UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Formulación matemática de la computación cuántica

Autor/es

MORAD BEN-LACHHAB-OUADRASSI LAAKEL

Director/es

EDUARDO SÁENZ DE CABEZÓN IRIGARAY

Facultad

Facultad de Ciencia y Tecnología

Titulación

Grado en Matemáticas

Departamento

MATEMÁTICAS Y COMPUTACIÓN

Curso académico

2019-20

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD
DE LA RIOJA

Facultad de Ciencia y Tecnología

Trabajo Fin de Grado

Grado en Matemáticas

Formulación matemática de la computación cuántica

Realizado por:
Morad Ben-Lachhab Laakel

Tutelado por:
Eduardo Sáenz De Cabezón Irigaray

Logroño, junio de 2020

Formulación matemática de la computación cuántica

Morad Ben-Lachhab Laakel

23 de junio de 2020

Resumen

Una buena forma de terminar una formación de grado en matemáticas es dejar abierta una puerta a uno de los muchos caminos interesantes que podemos encontrar a lo largo de la educación en la aplicación de las matemáticas. Este trabajo abre la puerta a una de las revoluciones tecnológicas que promete fuertes emociones en un futuro próximo.

Esta introducción a los fundamentos de la computación cuántica se divide en tres partes que nos llevan a un aprendizaje que va desde lo más básico de la materia hasta uno de los algoritmos más importantes que dio fama a esta manera de estudiar la computación.

En primer lugar, tenemos un capítulo que introduce el nuevo paradigma de computación. El uso de qubits (o quantum bits) en vez de los bits utilizados en la computación clásica, las puertas lógicas cuánticas tratadas como aplicaciones lineales y algunos de los algoritmos cuánticos más sencillos para mostrar el potencial de este tema. El segundo capítulo estudia uno de los ingredientes clave de los algoritmos más importantes de este nuevo paradigma, la transformada cuántica de Fourier. Y por último, se estudia desde su núcleo hasta un ejemplo de la implementación al famoso algoritmo de Shor, que factoriza en números primos números enteros de una manera mucho más eficiente que los análogos algoritmos clásicos.

Abstract

A good way to finish a mathematics degree is to leave a door open to one of the many interesting paths that we can find throughout education in the application of mathematics. This work opens the door to one of the technological revolutions that promises strong emotions in the near future.

This introduction to the fundamentals of quantum computing is divided into three parts that lead us to a learning process that goes from the basics of the subject to one of the most important algorithms that made this way of studying computing famous.

First, we have a chapter that introduces the new computing paradigm. The use of qubits (or quantum bits) instead of the bits used in classical computing, quantum logic gates treated as linear applications and some of the simplest quantum algorithms to show the potential

of this topic. The second chapter studies one of the key ingredients of the most important algorithms of this new paradigm, the Fourier quantum transform. And finally, we study from its core to an example of the implementation of Shor's famous algorithm, which factors into prime numbers integers in a much more efficient way than the classical analogous algorithms.

Introducción

Las computadoras que conocemos hoy en día (máquinas de Turing en el marco teórico y ordenadores, tablets, smatphones... en el marco práctico) están basados en la mecánica clásica, y en esta nos podemos encontrar limitaciones como que solo se pueda estar en un estado a la vez. Pero con los avances en el siglo XX de la mecánica cuántica, se ha demostrado que el mundo se comporta de manera bastante diferente.

La mecánica cuántica tiene la curiosa distinción de ser la materia que más interés suscita en el mundo científico y ser una de las más desconocidas. Comenzó a desarrollarse a principios del siglo XX, pero no es hasta los alrededores de 1980, cuando los pioneros de la computación cuántica comienzan a plantearse considerar como sistemas cuánticos los avances en la teoría de la computación y de la información. Los sistemas cuánticos nos dan la ventaja de poder utilizar propiedades como la superposición de estados al mismo tiempo o el entrelazamiento, propiedades que estudiaremos durante el trabajo. Basada en los principios de la mecánica cuántica, esta ciencia tiene como objetivo desarrollar algoritmos que sean significativamente más eficientes que los algoritmos en la computación clásica resolviendo el mismo problema.

Yuri Manin, Richard Feyman y Paul Benioff comenzaron a sugerir el desarrollo de ordenadores analógicos cuánticos, debido a que el coste computacional de la evolución de sistemas cuánticos en un ordenador clásico, era muy alto porque debía realizar muchas operaciones. En 1985, David Deutsch definió la máquina de Turing cuántica universal. El desarrollo prosiguió un poco más lento en los años venideros, Deutsch, Jozsa y Simon desarrollaron los primeros algoritmos y Bernstein y Vazirani desarrollaron la teoría de la complejidad cuántica. Pero a partir de la propuesta de Peter Shor, comenzó el aumento de la fama de esta materia. Y es que, Peter Shor sorprendió en 1994 descubriendo un eficiente algoritmo cuántico para el problema de la factorización entera y logaritmos discretos. Este hito desafió parte de la criptografía clásica, ya que podría romper sistemas como el RSA, y motivó los esfuerzos por realizar una criptografía cuántica, aunque aún no se han conseguido el número suficiente de qubits en un ordenador cuántico para dejar obsoletos los sistemas de cifrado.

Durante los siguientes años, se han anunciado la construcción de algunos ordenadores cuánticos (con 2 qubits, con 20, con 50), gracias al esfuerzo relativamente silencioso de empresas tecnológicas como IBM, Microsoft, Intel o Google. Y aunque para poder utilizar algoritmos clásicos en ordenadores

cuánticos se necesita desarrollar un algoritmo cuántico, cosa no trivial, lo que más obstaculiza el desarrollo de la materia es la construcción de los ordenadores. Estos funcionan a temperaturas cercanas al cero Kelvin (-273 °C), se necesitan superconductores y componentes para nada sencillos para el soporte y manipulación de los qubits, y añadido a este problema, tenemos la delicada sensibilidad de los qubits a las perturbaciones y el ruido.

De todas maneras, el potencial que promete esta tecnología a la mejora en la inteligencia artificial, la química, la simulación de sistemas cuánticos o a la resolución de problemas complejos entre otros, hace que las grandes empresas tecnológicas inviertan tiempo y dinero en obtener la supremacía en esta materia, aunque no se tengan todavía indicios de que se pueda utilizar de manera comercial para particulares, puede ser muy útil.

Este trabajo finaliza con el estudio del algoritmo de Shor debido a su importancia y a que el recorrido para llegar hasta él nos introduce de lleno en la computación cuántica.

Índice

1 Formalismo de la computación cuántica

1.1 Bits Cuánticos

El bit es el elemento fundamental de la computación clásica y es la unidad de medida de información que equivale a la selección entre dos alternativas de una manera determinista. En la computación cuántica se construye un concepto análogo al bit, el bit cuántico (o quantum bit en inglés) o también llamado para acortar, qubit. En esta sección se introduce este concepto, así como las propiedades que tiene un solo qubit y varios qubits comparandolos con las propiedades de los bits.

1.1.1 El qubit

Si nos preguntamos que es un qubit, deberíamos saber responder qué es matemática y físicamente. El objetivo de este trabajo es dar los fundamentos matemáticos de esta teoría, así pues, nos centraremos en los qubits como objetos matemáticos abstractos. La ventaja de tratarlos de esta manera es la libertad de poder construir una teoría general sobre la computación cuántica sin tener que adentrarnos en las dificultados que surgen a la hora de diseñar y construir un ordenador cuántico.

Así pues, si volvemos a preguntarnos qué es un qubit podemos responder diferenciándolo de su análogo el bit. El bit tiene un estado, el 0 o el 1, un quibit también tiene un estado que pueden ser $|0\rangle$ o $|1\rangle$ que corresponden a los estados 0 y 1 del bit, entonces ¿cuál es la diferencia?

Antes de responder a la pregunta vamos a introducir que significa la notación que vamos a utilizar, en el anexo 4, en la sección 4 encontramos algo más de información.

La notación de dirac, también conocida como notación bra-ket es la utilizada de manera estándar para describir los estados cuánticos en la teoría de la mecánica cuántica. Se llama así porque el producto interior de dos estados es denotado por $\langle\phi||\psi\rangle$, donde $\langle\phi|$ es el bra y $|\psi\rangle$ es el ket. En un espacio de Hilbert complejo $\mathbb{H}$, cada vector es un ket, $|\psi\rangle$. cada ket tiene un vector bra $\langle\phi|$ que pertenece al espacio de Hilbert dual $\mathbb{H}^*$, y representa su conjugado.

Vale, ahora que ya sabemos la notación que se va a utilizar en este trabajo vamos con la respuesta. La diferencia que hay entre los dos es que los qubits

no solo están en los estados $|0\rangle$ o $|1\rangle$, sino que pueden estar en los dos estados a la vez. Esto es posible porque se pueden hacer combinaciones lineales de los estados, también llamado superposición:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

Donde α, β, también conocidos como la amplitud, son números complejos, por tanto, la representación del estado del qubit anterior es la de un vector de 2 dimensiones en un espacio vetorial complejo. Los estados $|0\rangle$ o $|1\rangle$ se llaman estados básicos y forman una base ortonormal de este espacio vectorial.

Para el caso de los bits, al igual que hace un ordenador clásico, podemos determinar si se encuentra en el estado 0 o 1, en cambio, para el caso de los qubits no podemos determinar su estado cuántico. Osea, que el ordenador almacenará en su memoria un 0 o un 1 al final de la computación, por tanto, no determinará los valores de α y de β. Sin embargo, la mecánica cuántica nos permite llegar a más información sobre el estado cuántico, así, si medimos el qubit durante la computación obtendremos el estado 0 con probabilidad $|\ \alpha\ |^2$ y el estado 1 con probabilidad $|\ \beta\ |^2$, donde $|\ \alpha\ |^2 + |\ \beta\ |^2 = 1$. Esta condición además, se puede interpretar como que el estado de un qubit es un vector unitario en el espacio vectorial complejo.

Vale, es normal que llegados a este punto ya empecemos a tener dificultad en intuir estos sistemas cuánticos. La habilidad de un qubit a estar en superposición es un hecho que escapa de la manera común en que entendemos físicamente el mundo, así que para entenderlo, necesitamos profundizar en dos conceptos básicos de esta materia: la superposición y la medida.

- Consideremos un sistema físico que puede estar en N diferentes estados clásicos mutuamente excluyentes. Ahora, llamemos a estos estados $|0\rangle$, $|1\rangle$, ..., $|N-1\rangle$. Al referirnos a estados clásicos, nos referimos al estado en que se puede encontrar el sistema si lo observamos. Así un estado puramente cuántico $|\psi\rangle$ es una **superposición** de estados clásicos y se puede escribir como:

  $$|\psi\rangle = \alpha_0\,|0\rangle + \alpha_1\,|1\rangle + \cdots + \alpha_{N-1}\,|N-1\rangle.$$

  Donde α_i es un número complejo que representa que obtendremos el estado $|i\rangle$ con probabilidad $|\ \alpha_i\ |^2$ en $|\psi\rangle$. Por tanto, un estado cuántico puede estar en todos los estados clásicos al mismo tiempo. Matemáticamente, los estados $|0\rangle$, $|1\rangle$, ..., $|N-1\rangle$ forman una base ortonormal en

un espacio de *Hilbert* N-dimensional y por tanto, un estado cuántico $|\psi\rangle$ es un vector en este espacio.

- Supongamos que queremos medir el estado $|\psi\rangle$. En este caso no "veremos" la superposición de los estados, sino solo estados clasicos. Entonces si medimos el estado $|\psi\rangle$ solo veremos el estado $|i\rangle$, sin que sea uno en concreto, con la probabilidad de aparecer de $\mid \alpha_i \mid^2$, que es la norma de α_i ($\mid a + ib \mid = \sqrt{a^2 + b^2}$). Por tanto, observar un estado cuántico induce a una distribución de probabilidad de los estados clásicos con $\sum_{i=0}^{N-1} \mid \alpha_i \mid^2 = 1$, lo que nos lleva a que el vector $(\alpha_0, ..., \alpha_{N-1})$ tiene norma euclidea igual a 1.

Ahora pongamos un ejemplo para observar que un qubit esta entre ambos estados $|0\rangle$ y $|1\rangle$ y vamos a dar énfasis al hecho de que al medirlo nos va a dar la probabilidad que tiene de aparecer el estado $|0\rangle$ o el estado $|1\rangle$. Tenemos el siguiente qubit:

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

Si medimos este qubit, nos dará como resultado el estado 0 el cincuenta porciento de las veces ($\mid 1/\sqrt{2} \mid^2$), y el estado 1 el cincuenta por ciento restante. Este estado además, se sule denotar como $|+\rangle$. Su estado opuesto es el estado

$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

Vale, ahora dicho esto, podemos tener la idea un poco más clara de lo que el qubit representa ¿no?. Bueno, pero aún podemos dejarlo más claro dando una representación geométrica. Para ello volvamos a la representación de este qubit:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Como $\mid \alpha \mid^2 + \mid \beta \mid^2 = 1$, podemos reescribir la ecuación de la siguiente manera:

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right)$$

Donde θ, γ y φ son números reales. Como el factor $e^{i\gamma}$ podemos ignorarlo porque no tiene efectos observables, podemos escribir:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$$

Donde θ y φ son los puntos en la esfera unitaria de tres dimensiones, como se puede observar en la Figura 1. Esta esfera se suele llamar como la esfera de Bloch y nos da la idea intuitiva de como se visualiza el estado cuántico de un qubit. De todas formas, esto solo lo podemos mantener como una idea intuitiva, ya que no podemos abstraer esta idea fácilmente con múltiples qubit.

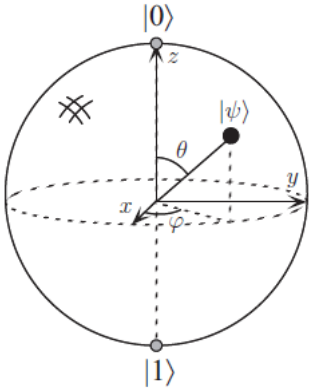


Figura 1: Esfera de Bloch

Así que, ¿Cuánta información representa un qubit? bueno, intentemos responderla con esta pregunta, ¿Cuántos puntos contiene la esfera unitaria? Infinitos, claro. Entonces ¿Podemos almacenar infinita infirmación en la infinita expansión binaria de θ? Pues parece que debido a cómo se comporta un qubit cuando se observa, no, ya que la medición nos dará solo uno de los dos estados clásicos y este comportamiento se debe a uno de los postulados fundamentales de la mecánica cuántica. Así que, aunque este en superposición el qubit cuando lo midamos, solo recibiremos un bit de información sobre el estado.

1.1.2 Múltiples qubits

Para poder hablar de múltiples qubits de una manera formal, primero vamos a verlo de una manera intuitiva.

Supongamos que tenemos dos qubits. Si estos dos qubits fuesen bits, tenemos 4 posibles estados clásicos, el 00, 01,10 y 11. Pues en su análogo el qubit, estos dos tienen cuatro estados básicos denotados como $|00\rangle$, $|01\rangle$, $|10\rangle$ y $|11\rangle$. Ahora que ya tenemos la idea vamos a dar una definición más formal. Asumamos lo siguiente:

Definición 1 *El estado de q qubits es un vector unitario en*

$$(\mathbb{C}^2)^{\otimes q} = \mathbb{C}^2 \otimes \cdots \mathbb{C}^2$$

Como ya hemos visto anteriormente, si tomamos la base canónica para $\mathbb{C}^2$, entonces, un único qubit se puede representar como

$$\alpha\,|0\rangle + \beta\,|1\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

donde α, $\beta\ \epsilon\ \mathbb{C}$ y $\mid \alpha \mid^2 + \mid \beta \mid^2 = 1$. Así dada la base canónica para $\mathbb{C}^2$, una base canónica de $(\mathbb{C}^2)^{\otimes q}$ es dada por :

$$|0\rangle_q = \underbrace{|0\rangle \otimes \cdots \otimes |0\rangle}_{q-veces}$$

$$|1\rangle_q = \underbrace{|0\rangle \otimes \cdots \otimes |0\rangle \otimes |1\rangle}_{q-veces}$$

$$|2^q - 1\rangle_q = \underbrace{|1\rangle \otimes \cdots \otimes |1\rangle}_{q-veces}$$

El estado de q qubits se puede representar como $|\psi\rangle = \Sigma_{j=0}^{2^q-1} \alpha_j\,|j\rangle_q$, con $\alpha_j\ \epsilon\ \mathbb{C}$ y $\Sigma_{j=0}^{2^q-1} \mid \alpha_j \mid^2$. Es muy importante notar que $(\mathbb{C}^2)^{\otimes q}$ es un espacio 2^q dimensional, lo que nos lleva a poder observar la diferencia con su análogo el bit. Dado un conjunto de q bits clásicos, su estado es una cadena de caracteres compuesta por $\{0,1\}^q$, por tanto es un espacio q dimensional. Por lo que la dimensión del estado de múltiples qubits crece exponencialmente en comparación con su análogo. Por tanto, la información que pueden llegar a registrar los qubits son 2^q números complejos y la de los bits q bits clásicos.

Aunque como ya hemos visto anteriormente, esto tiene una trampa, ya que si medimos el estado de múltiples qubits nos pasará lo mismo que con uno solo, no podremos acceder directamente al estado cuántico.

Ahora sigamos con la idea de un par de qubits. En un par de qubits también existe la superposición y por tanto este par puede ser representado como:

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle$$

Donde el primer 0 del estado $|00\rangle$ se refiere a que el estado del primer qubit es 0 y el segundo 0 se refiere a que el estado del segundo qubit es 0. Entonces, el 0 del estado $|01\rangle$ se refiere a que el estado del primer qubit es 0 y el 1 se refiere a que el estado del segundo qubit es 1.

Similar al caso para un único qubit, el resultado de medir este estado es que $x = 00, 01, 10, 11$ ocurre con probabilidad $\mid \alpha_x \mid^2$. También esta presente la condición de normalización $\Sigma_{x\epsilon\{0,1\}^2} \mid \alpha_x \mid^2$. En un sistema de múltiples qubits también podemos medir uno o varios qubits que pertenezcan al conjunto, por ejemplo, podemos decir que la medida de que el estado del primer qubit sea 0 es $\mid \alpha_{00} \mid^2 + \mid \alpha_{01} \mid^2$, obteniendo el siguiente estado después de la medición del primer qubit:

$$\left|\psi'\right\rangle = \frac{\alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle}{\sqrt{\mid \alpha_{00} \mid^2 + \mid \alpha_{01} \mid^2}}$$

Además, un sistema de dos qubits puede ser un estado producto o un estado entrelazado. Veamos primero las definiciones formales para dar después un ejemplo.

Definición 2 *Un estado cuántico* $|\psi\rangle\ \epsilon\ (\mathbb{C}^2)^{\otimes q}$ *se dice que se puede descomponer o factorizar, si puede ser expresado como un producto tensor* $|\psi_1\rangle \otimes \cdots |\psi_k\rangle$ *de* $k \geq 2$ *estados cuánticos sobre* $q_1, ..., q_k$ *con la propiedad de que* $q_1 + ... + q_k = q$.

Definición 3 *Un estado cuántico* $|\psi\rangle\ \epsilon\ (\mathbb{C}^2)^{\otimes q}$ *es un estado producto si se puede descomponer o factorizar en un producto tensor de estados de un solo qubit. En otro caso, es un estado entrelazado.*

Un ejemplo de un estado producto es el siguiente:

$$\frac{1}{2}\,|00\rangle + \frac{1}{2}\,|01\rangle + \frac{1}{2}\,|10\rangle + \frac{1}{2}\,|11\rangle$$

Ya que podemos factorizarlo en el siguiente producto tensor:

$$\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)$$

En el otro caso tenemos este estado:

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

que no se puede expresar como un producto de estados de un solo qubit y por tanto es un estado entrelazado.

Este estado, también denotado como $|\beta_{00}\rangle$, de dos qubits es el estado de Bell o también llamado estado EPR, en honor a Einsteins, Podolsky y Rosen quienes fueron los primeros en analizar estos estados y sus paradójicas propiedades. Este estado es importante por las muchas sorpresas que nos podemos encontrar en el. En primer lugar es el ingrediente clave de la teleportación cuántica, de la que hablaremos más adelante, es el prototipo de muchos otros estados interesantes y además tiene una propiedad muy interesante.

Supongamos que queremos medir la probabilidad de que el primer qubit tenga como estado el $|0\rangle$ o el $|1\rangle$. Para que esté en el estado $|0\rangle$ tenemos una probabilidad de $1/2$ obteniendo el estado $\left|\psi'\right\rangle = |00\rangle$, y lo mismo para el estado $|1\rangle$, una probabilidad de $1/2$ obteniendo el estado $\left|\psi'\right\rangle = |11\rangle$. Si volvemos a hacer lo mismo pero con el segundo qubit, observaremos que nos dará el mismo resultado que con el primer qubit, es decir, que los resultados de la medición están correlacionados. Aún aplicando algunas operaciones sobre este estado, volveremos a obtener la misma correlación sobre los resultados. Este hecho fue el que suscito gran interés después de que EPR señalaran estas extrañas propiedades en los estados como el estado de Bell. John Bell tomo estás ideas y las mejoró, e incluso demostró que las correlaciones en el estado de Bell son más fuertes de lo que podría existir entre sistemas clásicos.

Hay otros tres estados que también se denomina estados de Bell, que son los siguientes:

$$|\beta_{01}\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle$$
$$|\beta_{10}\rangle = \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle$$
$$|\beta_{11}\rangle = \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle$$

1.2 Puertas lógicas

Las operaciones o los cambios que pueden producirse en un estado cuántico se suelen llamar *puertas*, por su análogo el bit, cuyas operaciones o transformaciones son las puertas lógicas tales como AND, OR Y NOT.

1.2.1 Puertas lógicas elementales

Las puertas lógicas son usadas para manipular o convertir la información de un qubit. Consideremos el ejemplo de la puerta lógica de la computación clásica NOT. Esta, es usada para negar el estado, así, tenemos que $0 \to 1$ y que $1 \to 0$. ¿Podemos imaginar como puede ser el análogo para un qubit?

Bueno solo hay que pensar que queremos coger el estado $|0\rangle$ y transformarlo en el estado $|1\rangle$ o viceversa. Claro, pero ¿y qué pasa si nos encontramos con un estado en superposición? En este caso si escogemos el siguiente estado de un qubit:

$$\alpha\,|0\rangle + \beta\,|1\rangle$$

para negarlo deberíamos intercambiar la amplitud del estado $|0\rangle$ por la del $|1\rangle$, obteniendo:

$$\beta\,|0\rangle + \alpha\,|1\rangle$$

Por lo que nos encontramos que la puerta cuántica para la negación actúa linealmente sobre el estado de un qubit, y por tanto, podemos definir esta puerta lógica como una matriz con la siguiente forma:

$$X \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

donde podremos observar que si tomamos al estado $\alpha\,|0\rangle + \beta\,|1\rangle$ como un vector:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

y le aplicamos la transformación NOT, obtendremos

$$X \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$$

Otra de las puertas lógicas elementales es:

$$Z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

la cual deja al estado $|0\rangle$ inmutable y al estado $|1\rangle$ lo transforma al estado $-|1\rangle$.

Para completar tenemos dos matrices que junto con las otras dos anteriores forman las matrices de Pauli:

$$Y \equiv \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \text{ y la matriz identidad } I \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

que tienen la propiedad de que $XYZ = iI$.

Una de las puertas elementales más usadas en los algoritmos cuánticos es la transformación de *Hadamard*:

$$H \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

que es una rotación de un qubit transformando los estados $|0\rangle$ y $|1\rangle$ a dos estados en superposición, asi si aplicamos la puerta Hadamard, obtenemos:

$$H\left(\alpha|0\rangle + \beta|1\rangle\right) = \alpha\frac{|0\rangle + |1\rangle}{\sqrt{2}} + \beta\frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Lo que se puede interpretar como una rotación de $\dfrac{\pi}{2}$ sobre el eje y, y una rotación de π sobre el eje de las x. En la figura 2 se puede observar como actúa la puerta Hadamard.

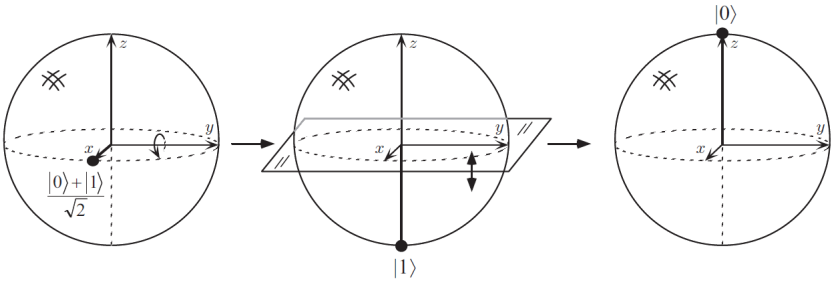


Figura 2: Esfera de Bloch con la puerta Hadamard

En la figura 2 podemos observar la siguiente operación:

$$H\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{|0\rangle + |1\rangle}{2} + \frac{|0\rangle - |1\rangle}{2} = |0\rangle$$

Otra puerta lógica es R_ϕ, la cual rota la fase del estado $|1\rangle$ en un ángulo de ϕ:

$$R_\phi \equiv \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

dejando el estado $|0\rangle$ inmutable.

Y por último, para el caso en el que queremos transformar dos qubits, tenemos la puerta CNOT:

$$CNOT \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

La cual niega el segundo qubit si el estado del primero es $|1\rangle$. En este caso podemos generalizar más, si consideramos la matriz U como una puerta lógica que transforma un qubit y consideramos que esta dentro de la siguiente puerta lógica que transforma dos qubits, podemos generalizar de manera que nos quede:

$$CU \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{11} & U_{12} \\ 0 & 0 & U_{21} & U_{22} \end{pmatrix}$$

Por último, tenemos la puerta lógica elemental que actúa en 3 qubits, se llama la puerta lógica *Toffoli* y se puede considerar como una CCNOT, ya que niega el tercer qubit si los dos primeros están en el estado 1.

Vistas ya las puertas, nos podemos hacer la siguiente pregunta, ¿la única razón por la que las puertas lógicas cuánticas son matrices es por la linealidad necesaria en los estados cuánticos? Parece que para responder deberemos recordar que la relación de las amplitudes en un estado cuántico debe cumplir que $|\ \alpha\ |^2 + |\ \beta\ |^2 = 1$. Así que si transformamos un estado cuántico $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ en otro $\left|\psi^{'}\right\rangle = \alpha^{'}\,|0\rangle + \beta^{'}\,|1\rangle$ este también debe de cumplir la condición $|\ \alpha^{'}\ |^2 + |\ \beta^{'}\ |^2 = 1$. Para ello, las puertas cuánticas deben cumplir la condición de ser matrices unitarias, es decir, que si U es una puerta lógica cuántica, debe cumplir que $U^tU = I$, donde U^t es la matriz adjunta de U (se obtiene a partir de transponer la matriz y conjugarla). Además, podemos llegar a la siguiente definición:

Definición 4 *Una operación aplicada por un ordenador cuántico con q qubits, también llamada puerta, es una matriz unitaria* $\mathbb{C}^{2^q \otimes 2^q}$.

1.2.2 Ejemplo: Teleportación cuántica

Vamos a da un ejemplo muy interesante en el que con la combinación de puertas lógicas elementales conseguimos hacer un algoritmo. En la siguiente sección veremos con más detalle los circuitos cuánticos y como representar los algoritmos.

El ejemplo que voy a mostrar es la teleportación cuántica, en la que tenemos dos partes, Alice y Bob. Alice quiere compartir con Bob un qubit de la forma $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ mediante el envío de bits clásicos. Esto sería imposible si Alice no conociese los pares EPR, asi escoge el siguiente par

$$|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

y lo "une" a su estado, obteniendo

$$|\psi_0\rangle = |\psi\rangle \otimes |\beta_{00}\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

$$= \frac{1}{\sqrt{2}}[\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|00\rangle + |11\rangle)]$$

En donde podemos observar que tenemos tres qubits, uno que viene del estado que quiere enviar Alice y otros dos que vienen del par EPR. Los dos primeros van a corresponder a Alice y el tercero a Bob. Obviamente ni Alice ni Bob conocen α o β del estado del qubit que se quiere enviar, así que, Alice y Bob se alejan para proceder a la teleportación.

El primer paso que aplica Alice es aplicar una puerta CNOT en sus dos qubits para entrelazarlos. Recordar que la puerta CNOT

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

toma el primer qubit como control y al segundo le aplica una NOT si el primero esta en el estado $|1\rangle$, así el estado se transforma en:

$$|\psi_1\rangle = CNOT|\psi_0\rangle$$

$$= \frac{1}{\sqrt{2}}[\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|10\rangle + |01\rangle)]$$

Después Alice aplica una transformación Hadamard, recordemos que es la matriz

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$$

esta puerta la aplica sobre el primer qubit, así obtenemos:

$$|\psi_2\rangle = H\,|\psi_1\rangle$$

$$= \frac{1}{\sqrt{2}}[\alpha\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)\left(|00\rangle + |11\rangle\right) + \beta\frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)\left(|10\rangle + |01\rangle\right)]$$

Si agrupamos por los qubits de Alice y de Bob obtendremos:

$$|\psi_2\rangle = \frac{1}{2}\underbrace{|00\rangle}_{A}\underbrace{\left(\alpha\,|0\rangle + \beta\,|1\rangle\right)}_{b} + \frac{1}{2}\,|01\rangle\left(\alpha\,|1\rangle + \beta\,|0\rangle\right) +$$

$$\frac{1}{2}\,|10\rangle\left(\alpha\,|0\rangle - \beta\,|1\rangle\right) + \frac{1}{2}\,|11\rangle\left(\alpha\,|1\rangle - \beta\,|0\rangle\right)$$

Así que por último Alice decide medir sobre sus dos qubits el estado $|\psi_2\rangle$, al medirlo, el estado colapsará a una de sus 4 opciones $(|00\rangle, |01\rangle, |10\rangle, |11\rangle)$ y Alice podrá mandarle dos bits cláscos a Bob. A su vez Bob, al recibir esos dos bits clásicos, solo tendra que observar el estado que corresponde a su tercer qubit, asi si recibe el estado 00, él tiene como estado $\alpha\,|0\rangle + \beta\,|1\rangle$, así solo le tendrá que aplicar la transformación identidad para obtener el estado que Alice quería teletransportar, con los demás estados pasa lo mismo, sólo que se aplican las distintas matrices de Pauli para conseguir el estado original de Alice:

- 01 corresponde con $\alpha\,|1\rangle + \beta\,|0\rangle$ así le aplicamos X y obtenemos $\alpha\,|0\rangle + \beta\,|1\rangle$
- 10 corresponde con $\alpha\,|0\rangle - \beta\,|1\rangle$ así le aplicamos Z y obtenemos $\alpha\,|0\rangle + \beta\,|1\rangle$
- 11 corresponde con $\alpha\,|1\rangle - \beta\,|0\rangle$ así le aplicamos XZ o iY y obtenemos $\alpha\,|0\rangle + \beta\,|1\rangle$

Así ya hemos conseguido la teleportación de un qubit de A a B, observar que el qubit en el lado de Alice ha sido destruido en lugar de copiarlo, además el hecho de copiar un qubit desconocido es imposible.

1.3 Circuitos cuánticos

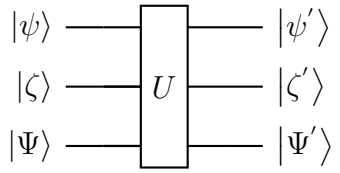
Todo lo referente a transformaciones en los estados de los qubits descritos anteriormente se pueden agrupar en un conjunto con el cual queremos llegar a un fin. Este conjunto puede ser descrito, al igual que en la computación clásica, en un un conjunto de cables y de puertas lógicas con el que podemos formar un algoritmo. A este conjunto se le llama circuito cuántico.
Es verdad que no es la única forma de representarlos, también se puede representar un algoritmo cuántico con una maquina de turing cuántica (el análogo a la maquina de Turing de la computación clásica), pero debido al alcance de este trabajo, solo hablaremos de circuitos cuánticos, entre otras cosas, porque son más populares actualmente.

En un *circuito clásico*, el ordenador computa paso por paso bits de manera Booleana mediante un grafo acíclico dirigido finito compuesto por las puertas lógicas AND, OR y NOT. Este tiene n nodos de entrada que contienen n bits de entrada, los cuales seran computados por el circuito que tendrá una o varias salidas en las que se obtendrá un valor, así la función que computa de manera booleana es de la forma $f : \{0,1\}^n \to \{0,1\}^m$.

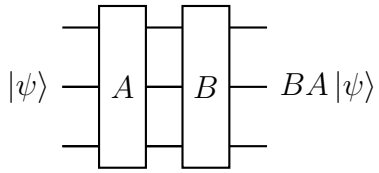
En un *circuito cuántico* se generaliza la idea del circuito clásico reemplazando las puertas lógicas clásicas por las puertas lógicas cuánticas elementales explicadas en la sección anterior, donde podemos aplicarlas en paralelo, tomando el producto tensor sobre ellas como operación, o secuencialmente, tomando productos ordinarios de matrices como operación. Cada qubit es representado mediante un cable que representa como se transporta la información hacia puertas lógicas que la transforman.

1.3.1 Notación para circuitos

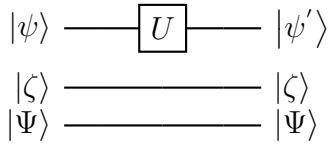
Supongamos que tenemos un circuito cuántico de 3 qubits al que se les aplica una puerta lógica U, este se dibujaría de la siguiente forma:



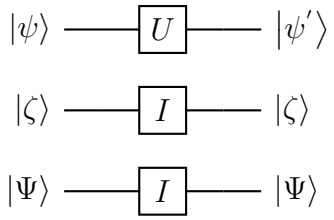
Notemos que los circuitos se leen de izquierda a derecha y que se van aplicando las matrices según van apareciendo, por eso si vemos este circuito:



Primero se le aplica la matriz unitaria A y luego la B. También se puede aplicar una puerta a un único qubit:



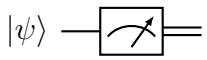
Lo que desde un punto de vista algebráico sería como aplicar una matriz unitaria $U\ \epsilon\ \mathbb{C}^{2\times 2}$, y una matriz Identidad del mismo espacio para los dos qubits restantes:



Ya que si agrupamos los tres qubits en un solo estado $|\sigma\rangle$, la aplicación de las tres matrices sería con su producto tensorial, $(I \otimes I \otimes U)|\sigma\rangle$.

Por último tenemos la puerta que representa la medida. Recordemos que la medida es el módulo de la amplitud correspondiente al estado al cuadrado. Se puede medir un solo qubit o varios a la vez, recordemos que los estados una vez medidos colapsan. La puerta de la medida se puede representar de varias maneras, en este trabajo se va a representar de la siguiente forma:

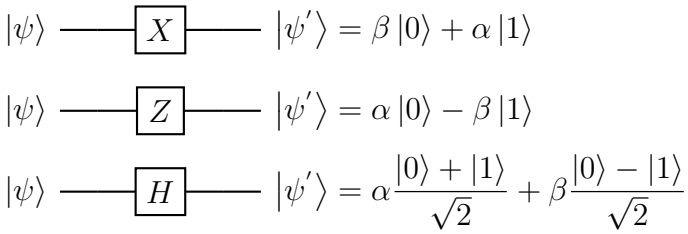
Para un estado de la forma $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, esta puerta convierte un qubit en un bit clásico probabilístico M (representado por los dos cables de la derecha) que es 0 con probabilida $\mid \alpha \mid^2$, o es 1 con probabilidad $\mid \beta \mid^2$:



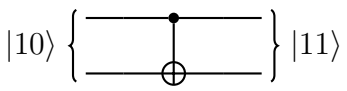
Veamos ahora, como se representan las puertas elementales en un circuito cuántico.

1.3.2 Representación de puertas elementales

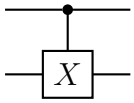
Recordemos que las puertas lógicas cuánticas elementales son las matrices de *Pauli*, la puerta R_θ, la transformación de *Hadamard*, la puerta CNOT y la CCNOT o puerta de *Toffoli*. La representación de puertas lógicas para un solo qubit es muy sencilla, consideramos el qubit con estado $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$:



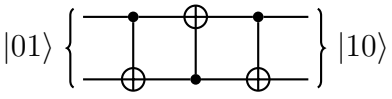
Después tenemos la puerta CNOT, en la que observaremos como escoge uno de los qubits como el qubit de control y al otro lo transforma si el de control esta en el estado $|1\rangle$, consideremos el siguiente estado:



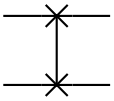
Que también es equivalente a:



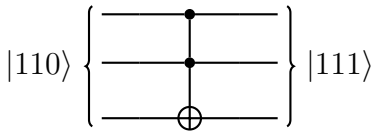
Una aplicación interesante de las puertas CNOT es el algoritmo que intercambia el estado de dos qubits, también llamada operación *SWAP*, la cual se representa de la siguiente manera:



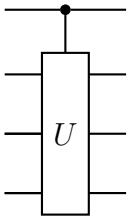
La cual también se puede escribir como:



La puerta *Toffoli* es muy parecida, con la diferencia de que toma dos qubits de control:

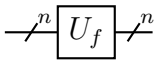


También tenemos la generalización de las puertas CNOT. Supongamos que tenemos una matriz unitaria U actuando sobre n qubits, así podemos definir una puerta *controlled-U* que es la expansión natural de la puerta CNOT. Esto se podría dibujar en una circuito de esta forma:



Donde la matriz U esta actuando sobre tres últimos qubit y el primero hace de qubit de control.

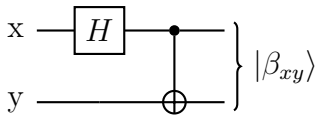
Por último, dada una función $f : \{0,1\}^n \to \{0,1\}^n$ la transformación que usualmente se llama U_f es la que transforma el estado $|x, y\rangle$ en el estado $|x, y \oplus f(x)\rangle$, donde $\oplus$ representa la suma módulo 2. Esta puerta se representa de la siquiente manera:



Donde la representación de $\not{}^n$ significa que se aplica para n qubits. Más adelante hablaremos de esta puerta y de sus utilidades.

1.3.3 Ejemplos de circuitos sencillos

Como primer ejemplo sencillo, vamos a considerar la combinación de una puerta Hadamard seguida de una puerta CNOT. Recordemos que una puerta Hadamard transforma un estado único en un estado en superposición, así, si escogemos el estado de dos qubits $|00\rangle$ la puerta Hadamard lo transformará en el estado $(|0\rangle + |1\rangle)|0\rangle / \sqrt{2}$ y la puerta CNOT tomará el primer qubit (que ahora esta en superposición) como qubit de control y transformará el estado en el estado $(|00\rangle + |11\rangle)/\sqrt{2}$, el cual podemos observar que es el estado de Bell $|\beta_{00}\rangle$. Este algoritmo cuántico se representa como un circuito de esta manera:



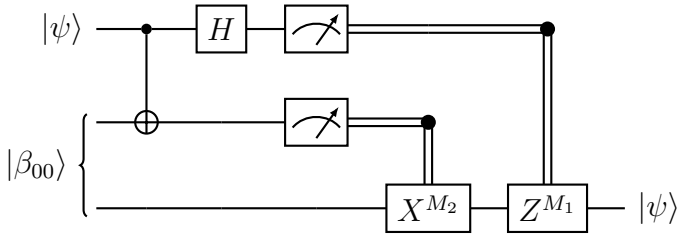
Notemos que la notación x, y y $|\beta_{xy}\rangle$ se muestra de esta manera ya que con este algoritmo podemos trasnformar el estado de dos qubits en cualquiera de los pares EPR de la siguiente manera:

$$|\beta_{xy}\rangle \equiv \frac{|0, y\rangle + (-1)^x |1, \bar{y}\rangle}{\sqrt{2}}$$

dando los siguientes resultado:

$$\begin{aligned}
|00\rangle &\longrightarrow |\beta_{00}\rangle \equiv \frac{|00\rangle + |11\rangle}{\sqrt{2}} \\
|01\rangle &\longrightarrow |\beta_{01}\rangle \equiv \frac{|01\rangle + |10\rangle}{\sqrt{2}} \\
|10\rangle &\longrightarrow |\beta_{00}\rangle \equiv \frac{|00\rangle - |11\rangle}{\sqrt{2}} \\
|11\rangle &\longrightarrow |\beta_{11}\rangle \equiv \frac{|01\rangle - |10\rangle}{\sqrt{2}}
\end{aligned}$$

Como segundo ejemplo sencillo vamos a ver como se dibujaría el circuito de la transportación cuántica explicado anteriormente. Recordemos que en este algoritmo tomábamos el qubit que Alice quería teletransportar y un estado de Bell. Para ponerlos en superposición aplicaba una puerta CNOT a los dos primeros qubits, después aplicaba una puerta Hadamard al primer qubit y mediamos el estado de los dos qubit que pertenecen a Alice. El algoritmo se representa de esta forma:



Donde M_1 y M_2 representan las medidas del primer y del segundo qubit respectivamente, así aplicaríamos la transformación adecuada dependiendo de los estados clásicos de los dos qubit medidos y de el qubit que pertenece a Bob. Recordemos que era:

- 00 corresponde con $\alpha |0\rangle + \beta |1\rangle$ así le aplicamos I $(X^0 Z^0)$ y obtenemos $\alpha |0\rangle + \beta |1\rangle$

- 01 corresponde con $\alpha|1\rangle + \beta|0\rangle$ así le aplicamos X (X^1Z^0) y obtenemos $\alpha|0\rangle + \beta|1\rangle$
- 10 corresponde con $\alpha|0\rangle - \beta|1\rangle$ así le aplicamos Z (X^0Z^1) y obtenemos $\alpha|0\rangle + \beta|1\rangle$
- 11 corresponde con $\alpha|1\rangle - \beta|0\rangle$ así le aplicamos XZ (X^1Z^1) y obtenemos $\alpha|0\rangle + \beta|1\rangle$

1.4 Ejemplos de algoritmos cuánticos

Los dos algoritmos cuánticos tachados como éxitos en esta materia son el algoritmo de factorización de *Shor* de 1994 (en el que nos adentraremos más adelante) y el algoritmo de búsqueda de *Grover* de 1996. En esta sección se presentan ejemplos de algoritmos que precedieron a estos dos, como el algoritmo de *Deutsch* y el algoritmo de $Deutsch - Jozsa$. Pero antes de nada vamos a hablar del paralelismo cuántico.

1.4.1 Paralelismo Cuántico

El paralelismo cuántico es una característica fundamental de muchos algoritmos cuánticos ya que es un efecto único que podemos usar en estos algoritmos y aun a riesgo de simplificarlo mucho, permite evaluar una función $f(x)$ para muchos valores de x simultáneamente. Supongamos que tenemos la función

$$f : \{0,1\}^n \to \{0,1\}^n$$

y que construimos un circuito cuántico con una transformación tal que transforme los siguientes estados de esta manera

$$|x\rangle|0\rangle \to |x\rangle|f(x)\rangle, \forall x \epsilon \{0,1\}^n.$$

Ahora, supongamos que aplicamos esta transformación a una superposición de todos los estados de los qubits x de entrada(consiguiendo la superposición aplicando n puertas Hadamard en paralelo sobre los qubits x), el resultado sería:

$$U\left(\frac{1}{\sqrt{2^n}}\sum_{x\epsilon\{0,1\}}|x\rangle|0\rangle\right) = \frac{1}{\sqrt{2^n}}\sum_{x\epsilon\{0,1\}}|x\rangle|f(x)\rangle$$

Donde podemos observar que aplicando una sola vez la transformación obtenemos una superposición de la evaluación de $f(x)$ sobre los 2^n valores de x. Además, si nos fijamos bien, esta transformación es la que anteriormente se

ha mencionado como U_f.

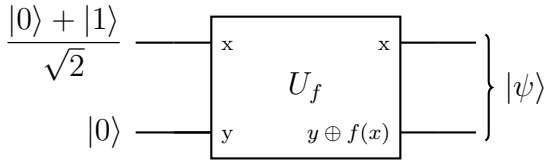
Esto es la generalización del siguiente ejemplo. Supongamos que tenemos la función booleana

$$f : \{0, 1\} \to \{0, 1\}$$

y la transformación U_f que transforma

$$|x, y\rangle \to |x, y \oplus f(x)\rangle$$

si el qubit y esta en el estado 0, el estado final después de la transformación será $f(x)$. Así, aplicamos una puerta Hadamard al estado $|0\rangle$ y con el siguiente circuito



conseguimos el siguiente resultado:

$$|\psi\rangle = \frac{|0, f(0)\rangle + |1, f(1)\rangle}{\sqrt{2}}$$

Sin embargo, esto por sí solo no es muy útil, ya que si medimos sobre la superposición solo nos dará un $|x_i, f(x_i)\rangle$ aleatorio y los demás se perderán. Como vamos a ver ahora, el paralelismo cuántico debe de combinarse para obtener mejores resultados.

1.4.2 Algoritmo de Deutsch

Volvemos al problema anterior en el que tenemos la función booleana $f : \{0, 1\} \to \{0, 1\}$ y la transformación U_f que transforma $|x, y\rangle \to |x, y \oplus f(x)\rangle$. Queremos saber si $f(x)$ es constante o equilibrada (mismo número de 0's que de 1's), así si queremos resolverlo de manera clásica solo tendríamos que evaluar dos veces la función para comprobarlo. Ahora veamos que con una simple modificación del anterior circuito cuántico con sólo una evaluación vamos a conseguir dar con el resultado. El algoritmo de *Deutsch* combina el paralelismo cuántico con una propiedad fundamental de la mecánica cuántica conocida como $interferencia$. La idea es controlar la probabilidad de que un sistema de qubits colapse en estados de medición particulares y esta propiedad nos permite sesgar la medición de un qubit hacia un estado o conjunto de estados deseados.

Comenzamos con el qubit de entrada

$$|\psi_0\rangle = |01\rangle$$

La forma de proceder para este algoritmo es que en vez de aplicar una puerta Hadamard tan solo al primer qubit, se aplica para los dos qubits, así obtenemos el siguiente estado:

$$|\psi_1\rangle = (H \otimes H)\,|\psi_0\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Lo siguiente es aplicar la transformación U_f sobre $|x\rangle\,(|0\rangle - |1\rangle)\,/\sqrt{2}$, para ello vamos a tener en cuenta estos pasos:

$$U_f(|0\rangle\,|0\rangle) = |0\rangle\,|0 \oplus f(0)\rangle = |0\rangle\,|f(0)\rangle$$

$$U_f(|0\rangle\,|1\rangle) = |0\rangle\,|1 \oplus f(0)\rangle = |0\rangle\,\overline{|f(0)\rangle}$$

$$U_f(|1\rangle\,|0\rangle) = |1\rangle\,|0 \oplus f(1)\rangle = |1\rangle\,|f(1)\rangle$$

$$U_f(|1\rangle\,|1\rangle) = |1\rangle\,|1 \oplus f(1)\rangle = |1\rangle\,\overline{|f(1)\rangle}$$

y también, teniendo esto en cuenta, vamos a seguir agrupando por $|0\rangle$ y por $|1\rangle$:

$$U_f(|0\rangle\,(|0\rangle - |1\rangle)) = |0\rangle\,\big|f(0) - \bar{f(0)}\big\rangle = |0\rangle\,(-1)^{f(0)}(|0\rangle - |1\rangle)$$

$$U_f(|1\rangle\,(|0\rangle - |1\rangle)) = |1\rangle\,\big|f(1) - \bar{f(1)}\big\rangle = |1\rangle\,(-1)^{f(1)}(|0\rangle - |1\rangle)$$

Así concretamos que la transformación tiene la forma de:

$$U_f(|x\rangle\,(|0\rangle - |1\rangle)) = |x\rangle\,(-1)^{f(x)}(|0\rangle - |1\rangle)$$

Por tanto, aplicando U_f sobre ψ_1 obtenemos una de estas dos posibilidades:

$$|\psi_2\rangle = U_f\,|\psi_1\rangle = \begin{cases} \pm\left(\dfrac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\dfrac{|0\rangle - |1\rangle}{\sqrt{2}}\right) & sif(0) = f(1) \\ \pm\left(\dfrac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\left(\dfrac{|0\rangle - |1\rangle}{\sqrt{2}}\right) & sif(0) \neq f(1) \end{cases}$$

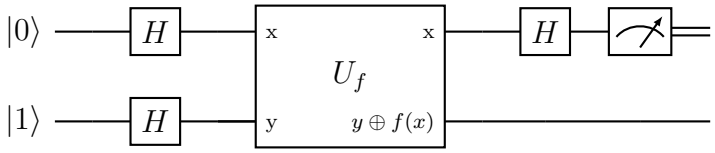
Por último, el algoritmo aplica una transformación Hadamard en el primer qubit, obteniendo:

$$|\psi_3\rangle = (I \otimes I \otimes H)\,|\psi_2\rangle = \begin{cases} \pm\,|0\rangle\left(\dfrac{|0\rangle - |1\rangle}{\sqrt{2}}\right) & sif(0) = f(1) \\ \pm\,|1\rangle\left(\dfrac{|0\rangle - |1\rangle}{\sqrt{2}}\right) & sif(0) \neq f(1) \end{cases}$$

observando que si $f(0) = f(1)$, $f(0) \oplus f(1) = 0$ y si $f(0) \neq f(1)$, $f(0) \oplus f(1) = 1$, podemos reescribir el resultado como:

$$|\psi_3\rangle = \pm\, |f(0) \oplus f(1)\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

midiendo finalmente el primer qubit determinaremos el resultado de $|f(0) \oplus f(1)\rangle$, si este valor es 1, la función es equilibrada, en cambio, si es 0, la función es constante. De esta manera el circuito tendra la siguiente forma:



Con este algoritmo, hemos pasado de una paralelismo cuántico que nos daba evaluaciones aleatorias de una función a poder determinar una propiedad global de la misma. Esto ya es un algoritmo cuántico que realiza menos iteraciones que un algoritmo clásico, ya que para resolver esto en un algoritmo clásico habría que evaluar dos veces la función. La clave de la diferencia es que un algoritmo clásico no puede excluir una de las dos alternativas en una sola iteración, en cambio, un algoritmo cuántico puede hacer que las dos alternativas interfieran entre sí para hallar una propiedad global de la función a determinar. Por eso es muy importante en estos algoritmos una elección inteligente de una transformación final que permita encontrar información global útil que no se pueda alcanzar rápidamente con los algoritmos clásicos.

1.4.3 Algoritmo de Deutsch-Jozsa

El algoritmo que se ha explicado anteriormente, es un caso simplificado de otro algoritmo cuántico más general, al que nos referiremos como algoritmo $Deutsch - Jozsa$. Este algoritmo fue propuesto por David Deutsch y Richard Jozsa en 1992 y fue uno de los primeros algoritmos diseñados para ser ejecutados por un ordenador cuántico. Como hemos visto en su caso simplificado (el algoritmo de $Deutsch$), este algoritmo ya es más eficiente que los algoritmos clásicos ya que aparte del paralelismo cuántico, aprovecha la interferencia, propiedad inherente de la superposición de estados cuánticos.

Supongamos que tenemos una función booleana $f : \{0,1\}^n \rightarrow \{0,1\}$, pero ahora vamos a explicarlo volviendo con Alice y con Bob. Alice, en madrid, selecciona un numero x entre 0 y 2^n - 1, y se lo manda por correo a Bob, que esta en Logroño. Bob define una función, constante o equilibrada (mismo número de ceros que de unos) y le envia el resultado, un cero o un

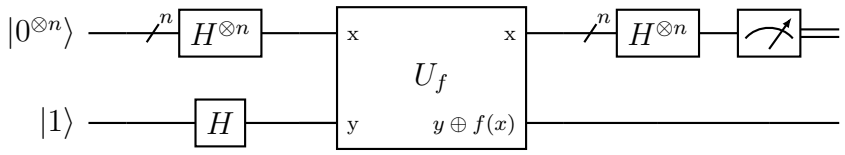
uno. El reto que propone Bob a Alice es que determine cuál de las dos funciones esta utilizando en el menor número de iteraciones. ¿Cuál es el mínimo?

Para un algoritmo clásico, Como Alice solo puede mandar un valor de x en cada carta, necesitará consultar al menos $2^{n-1}+1$ veces, ya que puede recibir 2^{n-1} ceros y obtener en la siguiente iteración un 1, y poder decir de manera determinista a Bob que está utilizando una función equilibrada, asi que hará $2^{n-1}+1$ consultas como mínimo. Pero si Alice tiene la oportunidad de intercambiar qubits en vez de bits con Bob, y Bob evalúa su función con una transformación unitaria U_f, entonces Alice podría llegar a su objetivo enviando una sola carta.

Esta es la generalización del algoritmo de *Deutsch*, Alice tiene un registro de n qubits para almacenar su consulta, y un qubit para que Bob almacene la respuesta. Alice comienza poniendo en superposición todos los qubits mediante transformaciones Hadamard y las envía a Bob. Bob recibe la información y evalúa $f(x)$ usando el paralelismo cuántico con una transformación unitaria U_f y almacena el resultado en el registro reservado para la respuesta. A continuación, Alice interfiere los estados en superposición usando transformaciones Hadamard en los qubits reservados para la consulta y finaliza realizando una medición adecuada para determinar si f es constante o equilibrada. Así el estado de los qubits de entrada es:

$$|\psi_0\rangle = |0\rangle^{\otimes n} |1\rangle$$

Y el circuito del algoritmo $Deutsch - Jozsa$ tiene la siguiente forma



Donde después de aplicar las puertas Hadamard obtenemos el estado:

$$|\psi_1\rangle = H^{\otimes n+1} |\psi_0\rangle = \sum_{x\epsilon\{0,1\}^n} \frac{|x\rangle}{\sqrt{2^n}} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

Donde observamos que los qubit reservados para la consulta estan en una superposición de todos los valores y el qubit reservado para la respuesta esta en una superposición entre los estados 0 y 1. Lo siguiente es evaluar la función f usando la transformada

$$U_f : |x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$$

obteniendo el siguiente estado(recordad las cuentas obtenidas en el algoritmo de *Deutsch*):

$$|\psi_2\rangle = U_f\,|\psi_1\rangle = \sum_x \frac{(-1)^{f(x)}\,|x\rangle}{\sqrt{2^n}}\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Ahora Alice tiene un conjunto de qubits que sirven como amplitud del resultado de la evaluación de la función de Bob

$$\sum_x \frac{(-1)^{f(x)}\,|x\rangle}{\sqrt{2^n}}$$

Ahora tiene que utilizar la propiedad de la interferencia sobre estos qubit en superposición usando transformaciones Hadamard. Para determinar el resultado de estas transformaciones, veamos que ocurre si aplicamos una puerta Hadamard sobre uno solo

$$H\,|x\rangle = \sum_{z\epsilon\{0,1\}} \frac{(-1)^{xz}\,|z\rangle}{\sqrt{2}}$$

Así, para los n qubit obtenemos,

$$H^{\otimes n}\,|x\rangle = \sum_{z\epsilon\{0,1\}^n} \frac{(-1)^{xz}\,|z\rangle}{\sqrt{2^n}}$$

donde xz es el producto interno bit por bit de x y z módulo 2. Usando esto, aplicando las transformaciones Hadamard obtenemos:

$$|\psi_3\rangle = H^{\otimes n}\,|\psi_2\rangle = \sum_z \sum_x \frac{(-1)^{xz+f(x)}\,|x\rangle}{2^n}\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Vamos a observar que la amplitud del estado $|0\rangle^{\otimes n}$ es

$$\sum_x \frac{(-1)^{f(x)}}{2^n}$$

así, si f es constante la amplitud de $|0\rangle^{\otimes n}$ es +1 o -1, dependiendo de que valor constante tome $f(x)$, ya que el módulo de $|\psi_3\rangle$ es unitario, de esto sigue que todas las demas amplitudes tienen que ser 0 y por tanto se obtiene (0,...,0) en la medida. En cambio, si f es equilibrada, los valores positivos y negativos de las amplitudes de $|0\rangle^{\otimes n}$ se cancelan, y al medir solo nos puede dar el valor 0. Resumiendo, si Alice al medir obtiene todo 0's, entonces la

función es constante, en otro caso es equilibrada.

Así el algoritmo de $Deutsch - Jozsa$ se puede resumir de la siguiente manera:

Algoritmo de Deutsch-Jozsa

Inputs:

- $f : \{0,1\}^n \to \{0,1\}$, con $x \epsilon \{0, \cdots 2^n - 1\}$ y f$f(x) \epsilon \{0,1\}$, siendo constante para todos los valores de x o equilibrada, esto es igual a 1 exactamente la mitad de los valores de x, y 0 la otra mitad de los valors.
- $U_f : |x, y\rangle \to |x, y \oplus f(x)\rangle$

Outputs: 0's si y solo si f es constante.

Ejecuciones: Una evaluación de U_f

Procedimiento

- Estado inicial
$$|\psi_0\rangle = |0\rangle^{\otimes n} |1\rangle$$
- Superposición de estados mediante transformaciones Hadamard
$$\sum_{x\epsilon\{0,1\}^n} \frac{|x\rangle}{\sqrt{2^n}} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$
- evaluación de la función f utilizando U_f
$$\sum_x \frac{(-1)^{f(x)} |x\rangle}{\sqrt{2^n}} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$
- Aplicar transformaciones Hadamard a todos los qubit excepto al último
$$\sum_z \sum_x \frac{(-1)^{xz+f(x)} |x\rangle}{2^n} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$
- Medir para obtener el output final.

2 Transformada de Fourier

Uno de los descubrimientos más espectaculares en la computación cuántica es que estos ordenadores pueden realizar algunas tareas que no son factibles en ordenadores que computan de manera clásica.
Por ejemplo, encontrar la factorización en números primos de un entero de n bits es un problema que actualmente esta atascado por no encontrar una manera eficiente de realizar este algoritmo, un ejemplo de este hecho es el sistema criptográfico de clave pública RSA, se basa en el problema de no poder factorizar números muy grandes. El algoritmo clásico de factorización con la mejor eficiencia demostrada requiere $O(2^{n/4}n^2)$ operaciones siendo n el número de bits, obtenido por Pollard y Strassen, aunque se estima que el algoritmo más eficiente requiere $e^{O(\sqrt[3]{n\log^2(n)})}$, obtenido por Lenstra, Manasse y Pollard. Esto nos lleva a la conclusión de que, al ver que es exponencial respecto al tamaño del número, a un ordenador clásico rápidamente se le hará imposible factorizar conforme vaya creciendo el número de bits.
Por el contrario, un algoritmo cuántico requiere realizar $O(n^2\log(n)\log\log(n))$ operaciones, lo que quiere decir, que un ordenador cuántico factoriza un número exponencialmente más rápido que los algoritmos clásicos más conocidos. Lo que nos puede llevar a pensar en ¿Qué otros problemas que no son factibles en la computación clásica podremos solucionar con la computación cuántica?

Una de las estrategias más usadas para resolver un problema en el mundo de las matemáticas o en las ciencias de la computación es la de transformar el problema en otro en el que se conozca la solución. Existen pocas transformaciones que aparezcan con tanta frecuencia y en contextos muy diferentes como la transformada que se desarrolla en este capítulo. Es uno de los ingredientes clave en el desarrollo de algoritmos cuánticos muy interesantes y es la transformada de Fourier cuántica. Antes de su desarrollo primero veamos el análogo clásico, la Transformada discreta de Fourier.

2.1 Transformada discreta de Fourier

La transformada de Fourier aparece en muchas disciplinas, incluso en muchas áreas de la computación, desde el procesamiento de señales a la compresión de datos y la teoría de la complejidad.
En la notación clásica del análisis de Fourier se refiere usualmente a la transformada de Fourier como una función, pero en dominios finitos una variante de esta notación es la que utilizaremos nosotros. Dado un vector $v \epsilon \mathbb{R}^N$ podemos tratarlo como una función v: $\{0,\ldots,N\} \to \mathbb{R}$ definido por $v(\mathrm{i}) = v_i$

donde de la transformada lineal $v \to F_N v = \hat{v}$ se llama transformada discreta de Fourier y F_N es la matriz asociada a la transformada discreta de fourier.

Así, para el propósito de este trabajo, la transformada de Fourier discreta (o DFT por sus siglas en inglés), es una matriz unitaria NxN donde todas sus entradas tendrán la misma magnitud. Así, para $N = 2$, tenemos una matriz que nos puede resultar familiar:

$$F_2 = H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

¡Exacto! Es la transformación de Hadamard que tanto se ha mencionado anteriormente, ya que la transformación de Hadamard pertenece al caso particular de $N = 2$.
Veamos ahora que para $N = 3$ no es posible con números reales, ya que no podemos formar 3 vectores ortogonales utilizando $\{+1, -1\}^3$. Por tanto, utilizaremos números complejos para definir la transformada de Fourier para cualquier N.

Definición 5 *Sea $\omega_N = e^{2\pi i/N}$ la N-ésima raíz de la unidad (donde raíz de la unidad es que para algún $\omega = e^{2\pi i/k}$, $\omega^k = 1$ para algún entero k) y sean j, $k \in \{0, \ldots, N-1\}$ las filas y las columnas respectivamente. Se define como la entrada (j, k) de la matriz F_N a $\frac{1}{\sqrt{N}}\omega_N^{jk}$, así la matriz tiene la siguiente forma:*

$$F_N = \frac{1}{\sqrt{N}} \begin{pmatrix} & \vdots & \\ \cdots & \omega_N^{jk} & \cdots \\ & \vdots & \end{pmatrix}$$

Notar que es sencillo comprobar que F_N es una matriz unitaria, para ello vamos a tener en cuenta esta proposición sobre la ortogonalidad de las raíces de la unidad:

Proposición 1 *Sean $p, q \in \{0, ..., N-1\}$, entonces*

$$\frac{1}{N} \sum_{j=0}^{N-1} \omega_N^{pj} \omega_N^{-qj} = \delta_{p,q} = \begin{cases} 1 & si\ p = q \\ 0 & en\ otro\ caso \end{cases}$$

Donde $\delta_{p,q}$ es la llamada delta de Kronecker.

Demostración. Si $p = q$, entonces $\omega_N^{p-q} = 1$ y se tiene:

$$\frac{1}{N}\sum_{j=0}^{N-1}\omega_N^{pj}\omega_N^{-qj} = \frac{1}{N}\sum_{j=0}^{N-1}(\omega_N^{p-q})^j = \frac{1}{n}\sum_{j=0}^{N-1}1 = 1$$

Si $p \neq q$ y como $p, q \,\epsilon\, \{0, ..., N-1\}$,entonces $\mid p-q \mid < N$, por eso N no divide a $p - q$ y $\omega_N^{p-q} \neq 1$. Aplicando la fórmula para la suma de una progresión geométrica obtenemos:

$$\frac{1}{N}\sum_{j=0}^{N-1}\omega_N^{pj}\omega_N^{-qj} = \frac{1}{N}\sum_{j=0}^{N-1}(\omega_N^{p-q})^j = \frac{1}{N}\frac{1-(\omega_N^{p-q})^N}{1-\omega_N^{p-q}} = \frac{1-1}{1-\omega_N^{p-q}} = 0$$

□

Con este resultado, si tomamos dos columnas cualesquiera k y k':

$$\sum_{j=0}^{N-1}\frac{1}{\sqrt{N}}(\omega_N^{jk}) * \frac{1}{\sqrt{N}}(\omega_N^{jk'}) = \frac{1}{N}\sum_{j=0}^{N-1}\omega_N^{j(k-k')} = \delta_{k,k'} = \begin{cases} 1 & \text{si } k = k' \\ 0 & \text{en otro caso} \end{cases}$$

Vemos que son ortogonales y por tanto F_N es una matriz unitaria. Al ser una matriz unitaria y simétrica tenemos que la inversa $F_N^{-1} = F_N^*$ y solo difiere de F_N por tener un signo negativo en el exponente de sus entradas.

Con esto tenemos que para cualquier vector $v \,\epsilon\, \mathbb{C}^N$, al vector $\hat{v} = F_N v$ se le llama transformada de Fourier de v, donde los elementos del vector son de la forma

$$\hat{v}_j = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}\omega_N^{jk}v_k$$

2.1.1 Transformada rápida de Fourier

Si calculásemos $\hat{v} = F_N v$ con $v \,\epsilon\, \mathbb{C}^N$ a partir de la definición, haciendo un producto entre una matriz y un vector, nos costaría $O(N)$ pasos (de sumas y multipilicaciones) por elemento del vector, y $O(N^2)$ pasos para calcular todos los elementos del vector $\hat{v}$. Sin embargo, uno de los hitos de la historia de la informática y de una importancia fundamental en el análisis matemático, es un algoritmo eficiente que calcula la transformada discreta de Fourier y su inversa, la llamada transformada rápida de Fourier (o FFT de sus siglas en inglés). Popularizada por James William Cooley y John Wilder Tukey en

1965, es de gran importancia en el tratamiento y filtrado digital de señales, resolución de ecuaciones en derivadas parciales o algoritmos de multiplicación rápida de grandes enteros. La importancia de este algoritmo reside en la diferencia operacional, ya que este algoritmo sólo necesita $O(N \log N)$ pasos, osea de una eficiencia operacional cuadrática a una eficiencia casi lineal.

Para nuestra definición, asumimos que $N = 2^n$, ya que podremos añadir ceros a nuestro vector para que su dimensión sea una potencia de dos. En general, depende de la factorización de N, pero se pueden dar transformadas rápidas de Fourier para cualquier N, incluso si es primo. La idea de la transformada es descomponer en transformadas más simples hasta llegar a transformadas de dos elementos, así, la clave de la FFT es reescribir las entradas de $\hat{v}$ de la siguiente manera:

$$\hat{v}_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} v_k$$

Ahora dividimos la suma en dos partes, la parte par con $k = 2m$ y la parte impar con $k = 2m + 1$:

$$\hat{v}_j = \frac{1}{\sqrt{N}} \left(\sum_{m=0}^{N/2-1} \omega_N^{j2m} v_{2m} + \sum_{m=0}^{N/2-1} \omega_N^{j(2m+1)} v_{2m+1} \right)$$

$$= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{N/2}} \sum_{m=0}^{N/2-1} \omega_{N/2}^{jm} v_{2m} + \omega_N^j \frac{1}{\sqrt{N/2}} \sum_{m=0}^{N/2-1} \omega_{N/2}^{jm} v_{2m+1} \right)$$

Así, estamos reescribiendo los elementos de la transformada de Fourier N-dimensional $\hat{v}$ en dos transformadas $N/2$-dimensionales que corresponden a la parte par de v y a la parte impar de v. Por tanto, si llamamos

$$E_j = \frac{1}{\sqrt{N/2}} \sum_{m=0}^{N/2-1} \omega_{N/2}^{jm} v_{2m}$$

a la parte par, y llamamos

$$O_j = \frac{1}{\sqrt{N/2}} \sum_{m=0}^{N/2-1} \omega_{N/2}^{jm} v_{2m+1}$$

a la parte impar, tenemos que

$$\hat{v}_j = \frac{1}{\sqrt{2}}(E_j + \omega_N^j \frac{1}{\sqrt{2}} O_j)$$

esta suma corresponde a la suma de las transformadas de Fourier E_j y de O_j de sus $N/2$ primeros términos. Para hallar los siguientes términos vamos a utilizar la periodicidad del ángulo complejo, tenemos que

$$\omega^{j+N/2} = e^{\frac{2\pi i(j+N/2)}{N}} = e^{\frac{2\pi ij}{N}} e^{\frac{2\pi i}{2}} = e^{\frac{2\pi ij}{N}} e^{\pi i} = -e^{\frac{2\pi ij}{N}} = -\omega^j$$

por tanto,

$$\hat{v}_{j+N/2} = \frac{1}{\sqrt{N}} \left(\sum_{m=0}^{N/2-1} \omega_N^{(j+N/2)2m} v_{2m} + \sum_{m=0}^{N/2-1} \omega_N^{(j+N/2)(2m+1)} v_{2m+1} \right)$$

$$= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{N/2}} \sum_{m=0}^{N/2-1} \omega_{N/2}^{jm} v_{2m} e^{2\pi mi} + \omega_N^j e^{\pi mi} \frac{1}{\sqrt{N/2}} \sum_{m=0}^{N/2-1} \omega_{N/2}^{jm} v_{2m+1} e^{2\pi mi} \right)$$

$$= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{N/2}} \sum_{m=0}^{N/2-1} \omega_{N/2}^{jm} v_{2m} - \omega_N^j \frac{1}{\sqrt{N/2}} \sum_{m=0}^{N/2-1} \omega_{N/2}^{jm} v_{2m+1} \right)$$

$$= \frac{1}{\sqrt{2}}(E_j - \omega_N^j \frac{1}{\sqrt{2}} O_j)$$

Y esta es la clave de la ganancia en la velocidad de los procesos que necesitamos para calcular la DFT de $\hat{v}$, obteniendo dos DFT menores de manera recursiva y combinando con $\frac{1}{\sqrt{2}} E_j$ y $\omega_N^j \frac{1}{\sqrt{2}} O_j$. Los tiempos $T(N)$ que necesitamos para calcular la FFT de $\hat{v}$ se calculan de manera que $T(N) = 2T(N/2) + O(N)$, ya que necesitamos calcular dos transformadas de dimensión $N/2$ mas $O(N)$ operaciones de adición para computar $\hat{v}$. De forma recursiva llegamos al tiempo $T(N) = O(N \log N)$, y de manera similar, calculamos con la misma eficiencia la transformada de Fourier inversa, ya que para la inversa solo hay que cambiar que $\frac{1}{\sqrt{N}} \omega_N^{-jk}$.

2.1.2 Multiplicación de dos polinomios

Una de las aplicaciones de la transformada rápida de Fourier es la de mejorar la eficiencia en la multiplicación de dos vectores.
Supongamos que tenemos dos polinomios $p[x]$, $q[x]$ ϵ $\mathbb{C}[x]$, de tal forma que

$$p(x) = \sum_{j=0}^{d} a_j x^j$$

y que

$$q(x) = \sum_{k=0}^{d} a_k x^k$$

y que queremos computar el producto de estos dos polinomios, de tal forma que

$$(p \cdot q)(x) = \left(\sum_{j=0}^{d} a_j x^j \right) \left(\sum_{k=0}^{d} a_k x^k \right) = \sum_{l=0}^{2d} \left(\sum_{j=0}^{2d} a_j b_{l-j} \right) x^l,$$

donde implícitamente podemos observar que $a_j = b_j = 0$ para $j > d$ y $b_{l-j} = 0$ si $j > l$. Si tomamos como coeficiente

$$c_l = \sum_{j=0}^{2d} a_j b_{l-j}$$

podemos observar que para computarlo tenemos que tomar $O(d)$ pasos de sumas y multiplicaciones, por tanto, para computar los valores de los coeficientes de $p \cdot q$ es necesario dar $O(d^2)$ pasos. Pero también es posible, usando la transformada rápida de Fourier, computarlo tomando $O(d \log d)$ pasos.

Para poder comenzar a dar una explicación de como usar el algoritmo de la FFT, primero tenemos que familiarizarnos con un concepto, el de la *convolución* de dos vectores.

Definición 6 *Dados dos vectores $a, b \epsilon \mathbb{C}^N$, la convolución de estos dos vectores es un vector $a * b \epsilon \mathbb{C}^N$ cuya l-entrada viene definada por*

$$(a * b)_l = \frac{1}{\sqrt{N}} \left(\sum_{j=0}^{N-1} a_j b_{l-j} + \sum_{j=0}^{N-1} a_j b_{N+l-j} \right)$$

Dado $m \epsilon \mathbb{Z}$, denotemos por m mod n el resto al dividir el número m entre n. Con esta notación vamos a escribir de manera más breve la convolución:

$$(a * b)_l = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j b_{(l-j) mod N}$$

Si de la definición tomamos $N = 2d + 1$ (el número de coeficientes no nulos de $p \cdot q$), hacemos el producto $(d + 1)$-dimensional anterior de los vectores a y b, y tomamos del producto polinomial

$$(p \cdot q)(x) = \left(\sum_{j=0}^{d} a_j x^j \right) \left(\sum_{k=0}^{d} a_k x^k \right) = \sum_{l=0}^{2d} \left(\sum_{j=0}^{2d} a_j b_{l-j} \right) x^l$$

el coeficiente

$$c_l = \sum_{j=0}^{2d} a_j b_{l-j}$$

veremos que el producto $p \cdot q$ es proporcional a las entradas de la convolución de $c_l = \sqrt{N}(a * b)_l$. Por tanto, a partir de aquí, es sencillo observar que los coeficientes de Fourier de la convolución coincide con el producto de los coeficientes de Fourier de a y de b. Esto nos lo da el teorema de convolución.

Teorema 1 *(de convolución). Sean a, b* ϵ $\mathbb{C}^n$*. Entonces*

$$F_n(a * b) = (F_n(a)) \cdot (F_n(b)),$$

o lo que es lo mismo

$$\widehat{(a * b)} = \hat{a} \cdot \hat{b}$$

Este teorema junto a su respectiva demostración en el anexo 4, inmediatamente nos sugiere el siguiente corolario, con el cual podremos hallar el algoritmo con el que computaremos el vector de coeficiente c_l:

corolario 1 *Sean a, b* ϵ $\mathbb{C}^n$*. Entonces*

$$a * b = F_n^{-1}((F_n(a)) \cdot (F_n(b)))$$

Así, si tomamos los vectores a y b y les aplicamos la FFT para obtener $\hat{a}$ y $\hat{b}$, multiplicamos estos dos vectores para obtener $\widehat{a * b}$, le aplicamos la inversa de la FFT para obtener $a*b$ y finalmente multiplicamos $a*b$ por $\sqrt{N}$, obtenemos el vector c de los coeficientes de $p \cdot q$. Por tanto, aquí ya conseguimos la eficiencia deseada, ya que la FFT y su inversa toma $O(N \log N)$ pasos y el producto escalar de dos vectores N-dimensionales toma $O(N)$ pasos, así este algoritmo tomara $O(N \log N) = O(d \log d)$ pasos. Debemos tener en cuenta que si se dan dos números a_d, ..., a_1, a_0 y b_d, ..., b_1, b_0 en notación decimal, podremos interpretarlos como los coeficientes de dos polinomios de grado d, p y q respectivamente:

$$p(x) = \textstyle\sum_{j=0}^{d} a_j x^j \text{ y } q(x) = \sum_{k=0}^{d} a_k x^k$$

Estos dos números seran la evaluación de los dos polinomios en $x = 10$, $p(10)$ y $q(10)$, y por tanto, el producto de estos dos números es la evaluación del producto de los dos polinomios en $x = 10$. Lo que intuitivamente podemos sugerir es utilizar el algoritmo anterior para conseguir multiplicar esos dos números de manera mas eficiente, $O(d \log d)$ pasos en vez de en $O(d^2)$ pasos como se computaría este producto utilizando la manera clásica de multiplicar. Sin embargo, si el objetivo es el de multiplicar dos números de d dígitos, y d no es un número muy grande, posiblemente te encuentres con que este algoritmo es menos eficiente que hacerlo de la manera tradicional, aunque aplicado a números grandes si que resulta la eficiencia de tomar $O(d \log d)$ pasos. Este algoritmo es conocido como el algoritmo Schönhage-Strassen (mejorado mas adelante por Fürer) y es uno de los ingredientes clave para el algoritmo de Shor.

2.2 La transformada cuántica de Fourier

Si nos ponemos desde la perspectiva de la teoría algoritmica, uno de los hitos efectivos en la ganacia exponencial de la computación cuántica es la transformada cuántica de Fourier (o QFT por sus siglas en inglés) que desarrollaremos en esta sección. La QFT es un algoritmo eficiente para el cálculo de la transformada de Fourier sobre amplitudes cuánticas, que aunque no ganemos con este algoritmo en la resolución de problemas clásicos, si que es el ingrediente clave en las aplicaciones como la *estimación de fase*, que es la aproximación de los valores propios de un operador unitario bajo ciertas circunstancias, para el algoritmo de búsqeda o *Algoritmo de Grover*, que nos permite resolver el algoritmo de búsqueda en una secuencia no ordenada de datos, o en el problema de la factorización o *Algoritmo de Shor* que veremos en el siguiente capítulo. En esta sección desarrollaremos el algoritmo para la QFT.

La transformada cuántica de Fourier es prácticamente la misma transformada que la transformada discreta de Fourier. Supongamos que tenemos la DFT definida anteriormente con F_N como la matriz NxN asociada, un vector cualquier $v \, \epsilon \, \mathbb{C}^N$, el vector $\hat{v} = F_N v$ es la llama transformada de Fourier discreta de v, donde los elementos del vector son de la forma

$$\hat{v}_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} v_k$$

Pues para la transformada de Fourier cuántica es exáctamente lo mismo, excepto por la notación. La QFT definida en una base ortonormal $|0\rangle$, ...,$|N-1\rangle$

se define como una operación lineal que actúa sobre el vector de amplitudes de un estado cuático de esta manera:

$$F_N \left|j\right\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} \left|k\right\rangle$$

donde recordemos que ω es una raíz N-ésima de la unidad ($\omega = e^{2\pi i/N}$) y j es una cadena de longitud n que representa la expresión binaria de un número entre 0 y 2^n-1. La matriz unitaria F_N asociada a esta transformación lineal es la misma definida anteriormente:

$$F_N = \frac{1}{\sqrt{N}} \begin{pmatrix} & \vdots & \\ \cdots & \omega_N^{jk} & \cdots \\ & \vdots & \end{pmatrix}$$

Aunque a simple vista, y a demás se ha dicho, parece que es la misma DFT, excepto por cambios en la notación, hay que tomar estas afirmaciones con cuidado y notar que existe una importante diferencia. Si bien para la DFT tomamos un vector v, que podemos escribir en un papel, y computamos su transformada $\hat{v} = F_N v$, también podemos escribir el resultado en un papel. Pero en el caso cuántico estamos trabajando con estados cuánticos, vectores de amplitudes que representan la superposición de estados. Así, para el caso N=2^n, donde es más simple crear un circuito unitario, lo que veremos a continuación es que la QFT la implementaremos en un circuito cuántico usando $O(n^2)$ puertas cuánticas elementales, lo que es exponencialmente más rápido incluso que la FFT, que utiliza $O(N \log N) = O(2^n n)$ operaciones, donde al computar obtendremos el vector de amplitudes del estado resultante y no las entradas de la transformada cuántica de Fourier escritos en un papel.

2.2.1 Ejemplos de QFT

Antes de implenmentar el circuito cuántico para la QFT, vamos a mostrar algunos ejemplos de como actúa la QFT sobre estados de un qubit. Tomamos $N = 2^n$

- **Para n = 1**. La matriz asociada a la QFT se puede calcular tomando $\omega_2^{0\cdot 0} = 1$, $\omega_2^{0\cdot 1} = 1$, $\omega_2^{1\cdot 0} = 1$ y $\omega_2^{1\cdot 1}$ = -1, que es la misma que la

transformación Hadamard, por tanto:

$$F_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

y

$$F_2 \left|0\right\rangle = \frac{1}{\sqrt{2}}(\left|0\right\rangle + \left|1\right\rangle)$$

- **Para n = 2**. Para calcular la matriz asociada tenemos que para $k = 0$ obtenemos $(\omega^{j \cdot 0}) = 1$ para todo j, para $k = 1$ obtenemos $(\omega^{j \cdot 1})_{j=0}^{j=3} =$ (1, i, -1, $-i$), para $k = 2$ obtenemos $(\omega^{j \cdot 2})_{j=0}^{j=3} =$ (1, -1, 1, -1) y para $k = 3$ obtenemos $(\omega^{j \cdot 3})_{j=0}^{j=3} =$ (1, $-i$, -1, i), Así la matriz que obtenemos es :

$$F_4 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

y los estados:

$$F_4 \left|0\right\rangle = \frac{1}{2}(\left|0\right\rangle + \left|1\right\rangle + \left|2\right\rangle + \left|3\right\rangle)$$

$$F_4 \left|1\right\rangle = \frac{1}{2}(\left|0\right\rangle + i\left|1\right\rangle - \left|2\right\rangle - i\left|3\right\rangle)$$

$$F_4 \left|2\right\rangle = \frac{1}{2}(\left|0\right\rangle - \left|1\right\rangle + \left|2\right\rangle - \left|3\right\rangle)$$

$$F_4 \left|3\right\rangle = \frac{1}{2}(\left|0\right\rangle - i\left|1\right\rangle - \left|2\right\rangle + i\left|3\right\rangle)$$

2.2.2 El circuito cuántico para la QFT

Para implementar correctamente un algoritmo eficiente para calcular la QFT, tomamos $N = 2^n$, donde $n \, \epsilon \, \mathbb{Z}$, y la base ortonormal $|0\rangle$,...,$|2^n - 1\rangle$, que es la base computacional para un ordenador cuántico de n qubits. Queremos lograr implementar la QFT con la operación lineal definida anteriormente con el estado base $|j\rangle$:

$$\left|j\right\rangle \rightarrow F_N \left|j\right\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} \left|k\right\rangle$$

Para ello, primero deberemos de ayudarnos de un cambio de notación, vamos a reescribir el estado $|j\rangle$ usando su expresión binaria $j = j_1 j_2 ... j_n$, o de una manera mas formal, $j = j_1 2^{n-1} + j_2 2^{n-2} + j_n 2^0$. Y para el estado $|k\rangle$ vamos

a hacer prácticamente lo mismo, tomamos su representación binaria $k = k_1k_2...k_n$ y la dividimos por 2^n, así tenemos $k/2^n = k_1/2^1 + k_2/2^2 + k_n/2^n$, su representación en fracción binaria. Con todo esto, reescribimos la QFT de manera que obtenemos:

$$F_{2^n} |j_1j_2...j_n\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi ijk/2^n} |k\rangle$$

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi ij \sum_{l=1}^{n} 2^{-l}k_l} |k_1k_2...k_n\rangle$$

por las propiedades de la exponencial obtenemos:

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \otimes_{l=1}^{n} e^{2\pi ijk_l2^{-l}} |k_l\rangle$$

cambiamos el productorio por el sumatorio:

$$= \frac{1}{2^{n/2}} \otimes_{l=1}^{n} \sum_{k_l=0}^{1} e^{2\pi ijk_l2^{-l}} |k_l\rangle$$

aplicamos la suma:

$$= \frac{1}{2^{n/2}} \otimes_{l=1}^{n} \left[|0\rangle + e^{2\pi ij2^{-l}} |1\rangle\right]$$

y aplicamos el producto:

$$= \frac{(|0\rangle + e^{2\pi i0.j_n} |1\rangle)(|0\rangle + e^{2\pi i0.j_{n-1}j_n} |1\rangle) \cdots (|0\rangle + e^{2\pi i0.j_1j_2\cdots j_n} |1\rangle)}{2^{n/2}}$$

Donde podemos notar que hemos tomado la fracción binaria $j2^{-n} = 0.j_1j_2 \cdots j_n$ y que $e^{0.j_n} \cdot e^{0.j_{n-1}} = e^{0.j_{n-1}j_n}$

Este producto nos muestra la manera de implementar un circuito eficiente para la transformada cuántica de Fouier. Este circuito contiene dos tipos de puertas lógicas distintos, estas son la transformación de Hadamard y la puerta lógica de rotación *Controlled-R_ϕ*:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \text{ y } R_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^\phi} \end{pmatrix}$$

Para ver como funciona el algoritmo, consideramos el estado $|j_1 j_2 \cdots j_n\rangle$, y vamos a aplicarle una transformación Hadamard al primer qubit:

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1} |1\rangle) |j_2 \cdots j_n\rangle$$

El siguiente paso es aplicar una puerta lógica *Controlled-*R_2 obteniendo el siguiente resultado:

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 j_2} |1\rangle) |j_2 \cdots j_n\rangle$$

Continuamos aplicando puertas lógicas *Controlled-*R_3, R_4, hasta R_n obteniendo:

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle) |j_2 \cdots j_n\rangle$$

El siguiente paso es aplicar los pasos anteriores de manera similar al segundo qubit, primero aplicamos una transformación Hadamard obteniendo:

$$\frac{1}{2}(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle)(|0\rangle + e^{2\pi i 0.j_2} |1\rangle) |j_3 \cdots j_n\rangle$$

Ahora aplicamos las puertas lógicas *Controlled-*R_2 hasta la R_{n-1} obteniendo el estado:

$$\frac{1}{2}(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle)(|0\rangle + e^{2\pi i 0.j_2 \cdots j_n} |1\rangle) |j_3 \cdots j_n\rangle$$

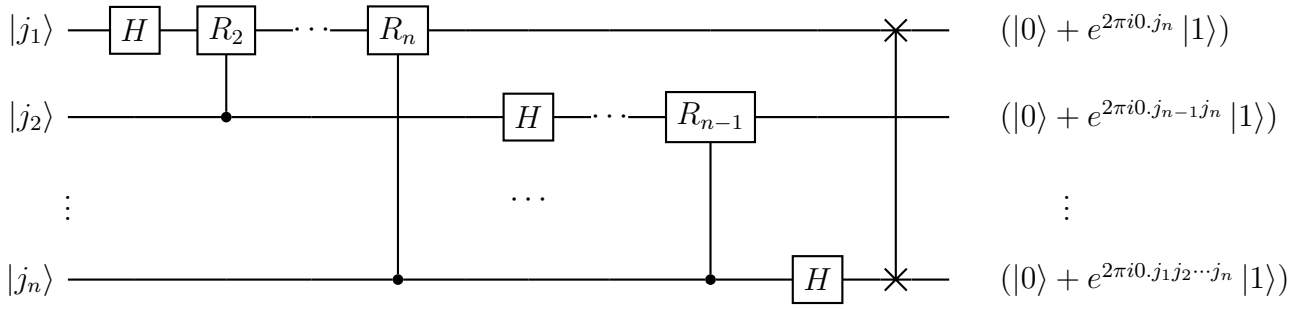
Continuando de esta manera con todos los qubit restantes, obtenemos un resultado similar al resultado final de operar linealmente la transformada cuántica de Fourier:

$$\frac{1}{2^{n/2}}(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle)(|0\rangle + e^{2\pi i 0.j_2 \cdots j_n} |1\rangle) \cdots (|0\rangle + e^{2\pi i 0.j_n} |1\rangle)$$

Utilizando operaciones SWAP, intercambiamos el orden de los qubit para obtener el resultado final:

$$\frac{1}{2^{n/2}}(|0\rangle + e^{2\pi i 0.j_n} |1\rangle)(|0\rangle + e^{2\pi i 0.j_{n-1} j_n} |1\rangle) \cdots (|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle)$$

Este procedimiento nos da el resultado deseado al aplicar la transformada cuántica de Fourier además de demostrar que es unitaria, ya que todas las transformaciones utilizadas son unitarias. El siguiente circuito representa el algoritmo eficiente descrito:



No se han incluido todas las puertas SWAP por limpieza en la representación del circuito, pero actúan en los estasdos j_l-j_{n-l+1} para $l \, \epsilon \, \{1, \cdots n\}$. Tampoco se ha incluido el factor de normalización $1/\sqrt{2}$ por lo mismo.

Ahora con el circuito ya a la vista, vamos a ver cuantas operaciones utiliza este algoritmo. En el primer qubit, hemos utilizado una transformación Hadamard y n - 1 puertas lógicas *Controlled-*R_ϕ, a continuación, en el segundo qubit, hemos utilizado una transformación Hadamard y n-2 puertas lógicas *Controlled-*R_ϕ, lo que acumula un total de n + (n - 1) operaciones. Continuando de esta manera, tenemos que para todos los qubits utilizamos $n + (n - 1) + \cdots + 1 = n(n + 1)/2$ operaciones, a lo que hay que añadir las operaciones SWAP, que son $n/2$ operaciones, lo que nos da un total de

$$\frac{n(n+1)}{2} + \frac{n}{2} = \frac{n(n+1)+n}{2} = O(n^2)$$

Por tanto, obtenemos que el algoritmo realiza en $O(n^2)$ operaciones para la transformada cuántica de Fourier. A diferencia del mejor algoritmo para calcular la transformada de Fourier discreta (algoritmo clásico), es la transformada rápida de Fourier (FFT), que utiliza $O(n2^n)$ operaciones, una diferencia exponencial con el algoritmo cuántico.
Finalmente, podemos obtener un circuito igual de eficiente para la transformada de Fourier inversa invirtiendo el orden de las operaciones y tomando la matriz adjunta de cada una de las matrices para obtener $F_N^{-1} = F_N^*$

A primera vista, este hecho es un hito para la computación, ya que la transformada de Fourier es un paso clave en muchas aplicaciones de procesado de datos. Por ejemplo, en el reconocimiento de fonemas, el primer paso es descomponer una señal en sus componentes, realizandose mediante la transformada de Fourier sobre la señal de sonido digitalizado. Lo que nos puede llevar a pensar que con este algoritmo podríamos acelerar el cálculo, desafortunadamente no hay ninguna forma conocida para esto. El problema

que nos encontramos es que no podemos acceder directamente a las amplitudes en una medición con el ordenador cuántico. Por lo tanto, no hay manera de acceder al vector de la transformada del estado original, y tampoco hay ningún algoritmo eficiente que prepare el vector de amplitudes original para ser transformado y obtenerlo. Así, la búsqueda de un uso eficiente del algoritmo de la QFT es más sutil de lo que se podía haber esperado. En el siguiente capítulo estudiaremos un algoritmo basado en las aplicaciones de la QFT.

Algoritmo para la transformada cuántica de Fourier

Inputs: Un estado de n-qubits.

Outputs: Vector de amplitudes del estado resultante.

Ejecuciones: n transformaciones Hadamard, $n(n$ - $1)/2$ puertas de rotación o *Controlled-*R_ϕ y $n/2$ operaciones SWAP. Un total de $(n(n+1)+n)/2$ operaciones.

Procedimiento:

- Estado inicial
$$|j_1 \cdots j_n\rangle$$
- Aplicación de una Transformación Hadamard y puertas lógicas *Controlled-*R_2 hasta R_{n-l+1} a cada qubit j_l con $l \epsilon \{1, \cdots n\}$.
$$\frac{1(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle)(|0\rangle + e^{2\pi i 0.j_2 \cdots j_n}|1\rangle) \cdots (|0\rangle + e^{2\pi i 0.j_n}|1\rangle)}{2^{n/2}}$$
- Aplicación de operaciones SWAP sobre los qubits j_l-j_{n-l+1} con $l \epsilon \{1, \cdots n\}$.
$$\frac{1}{2^{n/2}}(|0\rangle + e^{2\pi i 0.j_n}|1\rangle)(|0\rangle + e^{2\pi i 0.j_{n-1} j_n}|1\rangle) \cdots (|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle)$$

2.2.3 Ejemplo del algoritmo de la QFT para n=3

Para el ejemplo con $n = 3$, consideremos los siguientes tres qubit $|j_1 j_2 j_3\rangle$ y vamos a calcular su transformada mediante el algoritmo anteriormente descrito.

Primero aplicamos una transformación Hadamard al primer qubit obteniendo el primer resultado:

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1}|1\rangle)|j_2 j_3\rangle$$

Aplicamos las puertas *Controlled-*R_2 y R_3, y obtenemos:

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 j_2 j_3}|1\rangle)|j_2 j_3\rangle$$

A continuación, aplicamos una transformación Hadamard al segundo qubit, obteniendo:

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 j_2 j_3}|1\rangle)\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_2}|1\rangle)|j_3\rangle$$

Seguimos con una puerta lógica *Controlled-*R_2, obteniendo:

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 j_2 j_3}|1\rangle)\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_2 j_3}|1\rangle)|j_3\rangle$$

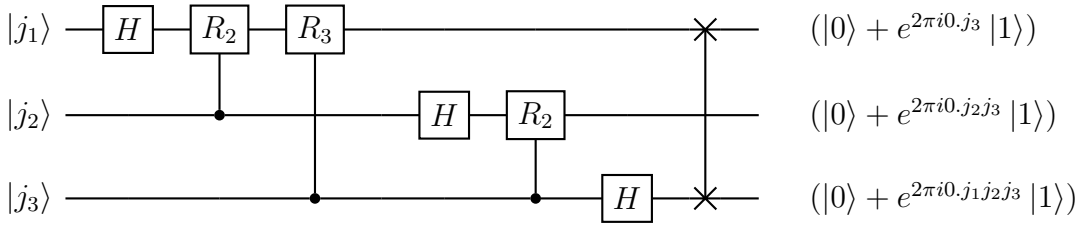
Por último, aplicamos una transformación Hadamard al tercer qubit:

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_1 j_2 j_3}|1\rangle)\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_2 j_3}|1\rangle)\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.j_3}|1\rangle)$$

Así lo último que queda es reordenar los qubit, esto es con una operación SWAP entre el qubit 1 y el 3 para obtener el resultado final:

$$F_8|j_1 j_2 j_3\rangle = \frac{(|0\rangle + e^{2\pi i 0.j_3}|1\rangle)(|0\rangle + e^{2\pi i 0.j_2 j_3}|1\rangle)(|0\rangle + e^{2\pi i 0.j_1 j_2 j_3}|1\rangle)}{\sqrt{2^3}}$$

El circuito para $n = 3$ tiene la siguiente forma:



Y para terminar, la matriz de la transformada cuyas entradas se definen como

$$F_8 = \frac{1}{\sqrt{8}}\sum_{j=1}^{3}\sum_{k=0}^{7}\omega_8^{jk}$$

donde $\omega_8 = e^{2\pi i/8}$, es la siguiente:

$$F_8 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega_8^1 & \omega_8^2 & \omega_8^3 & \omega_8^4 & \omega_8^5 & \omega_8^6 & \omega_8^7 \\ 1 & \omega_8^2 & \omega_8^4 & \omega_8^6 & 1 & \omega_8^2 & \omega_8^4 & \omega_8^6 \\ 1 & \omega_8^3 & \omega_8^6 & \omega_8^1 & \omega_8^4 & \omega_8^7 & \omega_8^2 & \omega_8^5 \\ 1 & \omega_8^4 & 1 & \omega_8^4 & 1 & \omega_8^4 & 1 & \omega_8^4 \\ 1 & \omega_8^5 & \omega_8^2 & \omega_8^7 & \omega_8^4 & \omega_8^1 & \omega_8^6 & \omega_8^3 \\ 1 & \omega_8^6 & \omega_8^4 & \omega_8^2 & 1 & \omega_8^6 & \omega_8^4 & \omega_8^2 \\ 1 & \omega_8^7 & \omega_8^6 & \omega_8^5 & \omega_8^4 & \omega_8^3 & \omega_8^2 & \omega_8^1 \end{pmatrix}$$

3 Algoritmo de Shor

En 1994, el matemático estadounidense Peter W. Shor, presentó uno de los primeros algoritmos relevantes que conseguía una reducción exponencial en el cálculo de los factores primos de un número N. Como bien se presentó en el capítulo anterior, el algoritmo clásico más eficiente en computación clásica que realiza este hecho, se estima que opera con una complejidad de $e^{O(\sqrt[3]{n \log^2(n)})}$. El algoritmo que presento Shor, consigue calcular este hecho en $O(\log(n)^2(\log\log(n))(\log\log\log(n)))$ operaciones, y por eso el algoritmo de Shor es uno de los algoritmos mas famosos de la computación cuántica. Por eso y por el transfondo que representa, ya que cualquier sistema criptográfico basado en el sistema RSA quedaría incapacitado con este algoritmo.

El problema de la factorización se enuncia de una manera muy sencilla:

Dado un número entero impar no primo N y con al menos dos factores primos distintos, ¿qué números primos multiplicados entre si dan como resultado este mismo número N?

Para dar respuesta a este enunciado, debemos encontrar los factores propios de este número N. Para ello debemos proceder de la siguiente manera:

Paso 1. Escogemos un entero aleatorio x tal que $1 < x < N$. Calculamos $d = \text{mcd}(x, N)$:

- Si $d > 1$, d es un factor de N.
- Si $d = 1$, procedemos al paso 2.

Paso 2. Calculamos el orden de x módulo N, es decir, el menor entero $r > 0$ tal que:

$$x^r \equiv 1 \pmod{N}$$

Paso 3.

- Si r es impar, volvemos al paso 1.
- Si r es par, procedemos al paso 4.

Paso 4.

- Si $x^{r/2} + 1 \equiv 0 \pmod{N}$, volvemos al paso 1.

- Si $x^{r/2} + 1 \neq 0 \pmod{N}$, procedemos al paso 5.

Paso 5. Calculamos $d = \text{mcd}(x^{r/2} + 1, N)$ y $d' = \text{mcd}(x^{r/2} - 1, N)$. Y así obtenemos d y d' que son factores no triviales de N

Este algoritmo puede computarse de manera clásica, pero es de manera cuántica como obtenemos la gran ventaja al poder calcular el paso 2 de una manera eficiente. Ahora nos vamos a adentrar en el corazón de las claves que nos encontramos en este algoritmo gracias a la aplicación de la QFT.

3.1 Estimación de fase

La transformada de fourier es la clave para un procedimiento general llamado estimación de fase, a la vez, este procedimiento es la llave para muchos algoritmos cuánticos funcionando como una subrutina en ellos. Supongamos un operador unitario U que tiene un vector propio $|u\rangle$ con un valor propio $e^{2\pi i\varphi}$ (donde $U|u\rangle = e^{2\pi i\varphi}|u\rangle$), y que el valor de $\varphi \, \epsilon \, [0, 1)$ es desconocido. Pues el objetivo del algoritmo de la estimación de fase es estimar el valor φ con t bits de precisión.

3.1.1 El algoritmo

El algoritmo utiliza dos registros. El primer registro contiene t qubits que inicialmente están en el estado $|0\rangle$. La elección de t depende del número de dígitos de precisión que queremos para estimar φ y de la probabilidad de éxito que deseamos para el algoritmo. El segundo registro comienza en el estado $|u\rangle$, y contiene el número de qubits necesario para almacenar $|u\rangle$. El algoritmo se compone de dos partes. La primera parte comienza aplicando una transformación Hadamard al primer registro, seguida de la aplicación de puertas *controlled-U* en el segundo registro, con el operador U elevado a las sucesivas potencias de dos. Así, si aplicamos la transformación hadamard al primer registro nos queda:

$$\frac{(|0\rangle + |1\rangle)}{\sqrt{2}} \cdots \frac{(|0\rangle + |1\rangle)}{\sqrt{2}}$$

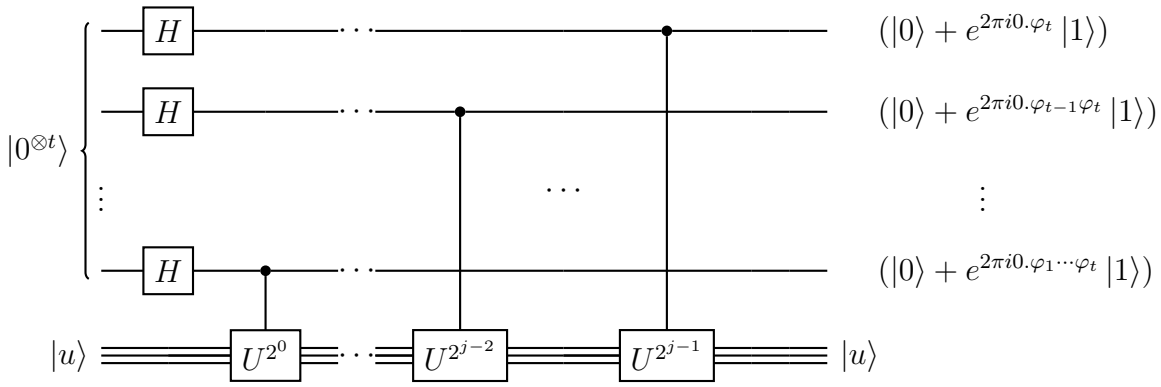
y después al aplicar los operadores U^{2^j} el estado final de $|0 \cdots 0\rangle$ que nos encontramos es:

$$\frac{1}{\sqrt{2^t}}(|0\rangle + e^{2\pi i 2^{t-1}\varphi}|1\rangle)(|0\rangle + e^{2\pi i 2^{t-2}\varphi}|1\rangle) \cdots (|0\rangle + e^{2\pi i 2^0 \varphi}|1\rangle)$$

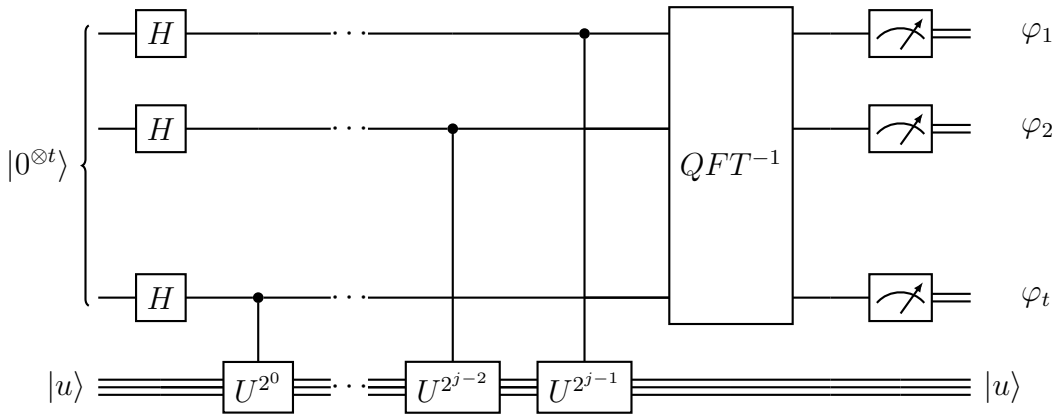
donde para facilitar la visión del algoritmo vamos a suponer que φ expresado con t bits de precisión es $\varphi = 0.\varphi_1 \cdots \varphi_t$. Así el resultado anterior se puede reescribir como:

$$\frac{1}{\sqrt{2^t}}(|0\rangle + e^{2\pi i 0.\varphi_t}|1\rangle)(|0\rangle + e^{2\pi i 0.\varphi_{t-1}\varphi_t}|1\rangle)\cdots(|0\rangle + e^{2\pi i 0.\varphi_1\varphi_2\cdots\varphi_t}|1\rangle)$$

Y el segundo registro se mantiene en el mismo estado $|u\rangle$. Así, la representación del circuito cuántico de esta primera parte es la siguiente:



La segunda parte del algoritmo corresponde con la aplicación de la transformada cuántica de Fourier inversa al primer registro, recordemos que dicho algoritmo se obtiene a partir de invertir el de la transformada cuántica de Fourier visto en el anterior capítulo y que se realiza en $O(t^2)$ operaciones. Que apliquemos la QFT inversa es bastante intuitivo, ya que si observamos el resultado anterior de aplicar la primera parte del algoritmo de estimación de fase, podemos ver que es el resultado de aplicar la QFT a un estado $|\varphi_1\varphi_2\cdots\varphi_t\rangle$. !Pues ese es el resultado de aplicar la QFT inversa! Por tanto, la tercera parte del algoritmo es la de medir cada uno de los estados del primer registro, lo que nos dara la estimación de φ con t bits de precisión. El circuito del algoritmo se quedaría de la siguiente forma:



Para finalizar, observemos que lo esencial de este algoritmo es la transformación lineal operada por la QFT:

$$\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi i \varphi j} |j\rangle |u\rangle \rightarrow |\hat{\varphi}\rangle |u\rangle$$

Donde $|\hat{\varphi}\rangle$ es, después de ser medido, una estimación de φ.

3.1.2 Requisitos

Si se da el caso ideal de que φ puede ser descrito en exactamente t bits, con el algoritmo de la estimación de fase obtendríamos exactamente φ. En el caso en el que no obtenemos con exactamente t bits la expresión binaria de φ lo que obtendremos es una buena aproximación de φ con una alta probabilidad de éxito. Para obtener una buena aproximación es necesario realizar el siguiente procedimiento.
Sea b un entero perteneciente a $[0, 2^t - 1]$ donde $b/2^t = 0.b_1 \cdots b_t$ es la mejor aproximación de φ y que satisface que $0 \leq \delta = \varphi - b/2^t \leq 2^{-t}$. El objetivo es mostrar como el algoritmo produce un resultado cercano a b y que es una estimación de φ bastante aproximada. Comenzamos con el estado

$$\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi i \varphi j} |j\rangle$$

le aplicamos la QFT inversa que ya sabemos es la operación lineal

$$|j\rangle \rightarrow \frac{1}{\sqrt{2^t}} \sum_{l=0}^{2^t-1} e^{-2\pi i j l/2^t} |l\rangle$$

y obtenemos el estado

$$\frac{1}{\sqrt{2^t}}\frac{1}{\sqrt{2^t}}\sum_{l=0}^{2^t-1}\sum_{j=0}^{2^t-1} e^{-2\pi ijl/2^t} e^{2\pi i\varphi j}\,|l\rangle$$

Sea α_l la amplitud de $|(b+l)(\text{mod } 2^t)\rangle$, este estado se define como

$$\alpha_l = \frac{1}{2^t}\sum_{j=0}^{2^t-1} e^{-2\pi ij(l+b)/2^t} e^{2\pi i\varphi j}\,|l\rangle = \frac{1}{2^t}\sum_{j=0}^{2^t-1}\left(e^{2\pi i(\varphi-(l+b)/2^t}\right)^k$$

Y como es una serie geométrica y sabiendo que $\delta = \varphi - b/2^t$ obtenemos:

$$\alpha_l = \frac{1}{2^t}\left(\frac{1-e^{2\pi i(2^t\varphi-(b+l))}}{1-e^{2\pi i(\varphi-(b+l)/2^t)}}\right) = \frac{1}{2^t}\left(\frac{1-e^{2\pi i(2^t\delta-l)}}{1-e^{2\pi i(\delta-l/2^t)}}\right)$$

Como $\mid 1-e^{i\theta} \mid \leq 2$ podemos acotar de esta manera:

$$\alpha_l \leq \frac{1}{2^t}\left(\frac{2}{\mid 1-e^{2\pi i(\delta-l/2^t)} \mid}\right)$$

Y además sabiendo que geométricamente

$$1-e^{i\theta} = -e^{i\theta/2}(e^{i\theta/2}-e^{-i\theta/2}) = -2ie^{i\theta}\sin\frac{\theta}{2}$$

para $0 \leq \theta \leq 2\pi$, tenemos que el $\sin\theta/2 \geq 0$ en este intervalo y que por lo tanto el módulo es $2\sin\theta/2$. Por tanto tenemos que $\mid 1-e^{i\theta} \mid = 2\sin \mid \theta \mid$, y como $\sin x$ es una función cóncava en $0 \leq x \leq \pi/2$ entonces $\sin x \geq 2x/\pi$ en este intervalo y podemos acotar y obtener

$$\mid 1-e^{i\theta} \mid \geq \frac{2 \mid \theta \mid}{\pi}$$

En el intervalo $-\pi \leq \theta \leq \pi$. Ahora modificando el intervalo a nuestro contexto tenemos que $-2^{t-1} < l < 2^{t-1}$ y por tanto $-\pi \leq 2\pi(\delta - l/2^t) \leq \pi$, así

$$\mid \alpha_l \mid \leq \frac{1}{2^{t+1}(\delta - l/2^t)} = \frac{1}{2(2^t\delta - l)}$$

Ahora, supongamos que la medición final es m, por tanto el objetivo es acotar la probabilidad de que el valor de m sea $\mid m-b \mid > E$, donde E es un entero positivo que revela nuestra tolerancia de error. Así la probabilidad de pasarnos de un error ϵ es:

$$p(\mid m-b \mid > E) \leq \sum_{2^{t-1}<l\leq-(E+1)} \mid \alpha_l \mid^2 + \sum_{(E+1)\leq l\leq 2^{t-1}} \mid \alpha_l \mid^2$$

$$= \frac{1}{4}\left(\sum_{l=-2^{t-1}+1}^{-(E+1)} \mid \frac{1}{(l-2^t\delta)^2} \mid + \sum_{l=(E+1)}^{-2^{t-1}} \mid \frac{1}{(l-2^t\delta)^2} \mid\right)$$

Sabiendo que $0 \leq 2^t\delta \leq 1$, obtenemos:

$$p(\mid m-b \mid > E) \leq \frac{1}{4}\left(\sum_{l=-2^{t-1}+1}^{-(E+1)} \mid \frac{1}{(l-1)^2} \mid + \sum_{l=(E+1)}^{-2^{t-1}} \mid \frac{1}{(l-1)^2} \mid\right)$$

y como en el primer sumatorio el índice es negativo tenemos:

$$p(\mid m-b \mid > E) \leq \frac{1}{4}\left(\sum_{l=-2^{t-1}+1}^{-(E+1)} \mid \frac{1}{l^2} \mid + \sum_{l=(E+1)}^{-2^{t-1}} \mid \frac{1}{(l-1)^2} \mid\right)$$

donde si a $l^{'} = (l-1)$ y $l^{''} = -l$ obtenemos

$$\leq \frac{1}{4}\left(\sum_{l=(E+1)}^{2^{t-1}+1} \mid \frac{1}{(l^{''})^2} \mid + \sum_{l=(E+1)}^{-2^{t-1}} \mid \frac{1}{(l^{'})^2} \mid\right)$$

que al ser el mismo sumatorio podemos acotar como:

$$\leq \sum_{l=E}^{2^{t-1}} \mid \frac{1}{l^2} \mid$$

que se puede aproximar como una integral:

$$\leq \int_{E-1}^{2^{t-1}-1} dl \frac{1}{l^2} = \frac{1}{2(E-1)}$$

Donde suponemos que t es un número muy grande. Así la probabilidad de que $m-b$ sea menor que E es:

$$p(\mid m-b \mid < E) = 1 - \frac{1}{2(E-1)}$$

Entonces, si queremos aproximar φ con una precisión de 2^{-n}, tomamos $E = 2^{t-n} - 1$. Ahora hacemos el cambio de variable $t = n + p$, y sustituyendo en la probabilidad de conseguir un error menor a E obtenemos:

$$p(\mid m-b \mid < E) = 1 - \frac{1}{2(2^p-2)}$$

Y como queremos una probabilidad de éxito cercana a 1:

$$p(\mid m-b \mid < E) = 1 - \frac{1}{2(2^p-2)} = 1 - \epsilon$$

Para ello, tomamos

$$p = \log\left(2 + \frac{1}{2\epsilon}\right) \text{ y } t = n + \log\left(2 + \frac{1}{2\epsilon}\right)$$

Por tanto, esta es la t que debemos elegir para aproximar φ con n bits de precisión y una probabilidad de éxito igual a 1 - ϵ.

El algoritmo de estimación de fase es un algoritmo interesante ya que resuelve un problena no trivial como estimar el valor propio de un vector propio dado de una matriz unitaria. Pero además, otro hecho que lo hace tan interesante es el de que algunos problemas se pueden reducir a la estimación de fase, tal y como veremos a continuación.

Algoritmo de estimación de fase

Inputs: Puertas lógicas *controlled-U^j*, para $j \in \mathbb{Z}$, un vector propio $|u\rangle$ de U con un valor propio $e^{2\pi i\varphi_u}$ y $t = n + (\log(2 + 1/2\epsilon))$ qubits inicialmente en el estado $|0\rangle$.

Outputs: Una aproximación $\hat{\varphi_u}$ con n bits de precisión de φ_u.

Ejecuciones: $O(t^2)$ operaciones y una llamada a cada puerta lógica *controlled-U^j* con una probabilidad de éxito de 1 - ϵ.

Procedimiento:

- Estado inicial $|0\rangle |u\rangle$
- Creamos superposición en el registro $|0\rangle$ y aplicamos las puertas lógicas *controlled-U^j*:

$$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle |u\rangle = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi i j\varphi_u} |j\rangle |u\rangle$$

- Aplicamos la transformada cuántica de Fourier inversa:

$$\rightarrow |\hat{\varphi_u}\rangle |u\rangle$$

- Medimos el primer registro:

$$\hat{\varphi_u}$$

3.2 Algoritmo de cálculo del orden

Como ya hemos dicho anteriormente, el algoritmo de estimación de fase se utiliza para resolver varios problemas interesantes. En este caso vamos a uno de los problemas más importantes, el algoritmo de cálculo del orden. Este es el algoritmo que resuelve la parte cuántica del algoritmo de Shor.

Supongamos dos enteros positivos x y N, donde $x < N$, tomemos $\mathbb{Z}_N = \{0, \cdots, N-1\}$ donde si también asociamos las operaciones de adición y multiplicación módulo N, formamos un anillo. Para denotar el grupo multiplicativo $\mathbb{Z}_N^*$ utilizaremos:

$$\mathbb{Z}_N^* = \{a \epsilon \mathbb{Z}_N : mcd(a, N) = 1\}.$$

El problema que queremos resolver, es el de calcular el orden de un elemento $x \epsilon \mathbb{Z}_N^*$, donde el orden se define como el menor entero positivo r tal que

$$x^r \equiv 1 \pmod{N}$$

Calcular el orden de un elemento mediante algoritmos clásicos es un procedimiento que requiere muchas operaciones, en cambio, utilizando la estimación de fase para resolverlo utilizaremos $O(n)$ operaciones, donde $n = \log N$ es el número de bits necesario para describir N en binario.

3.2.1 Transformando el problema

El algoritmo de cálculo del orden es el algoritmo de estimación de fase aplicando el operador unitario

$$U |y\rangle \equiv |xy \pmod{N}\rangle$$

con $y \epsilon \{0,1\}^l$, donde también debemos notar que para $N \leq y \leq 2^n - 1$ utilizamos para simplificar que $U |y\rangle \equiv |y\rangle$, por tanto la transformación unitaria U únicamente actúa de manera no trivial cuando $0 \leq y \leq N-1$.

Para poder habilitar la utilización del algoritmo de estimación de fase necesitamos dos requerimientos importantes: necesitamos tener procedimientos eficientes para implementar las operaciones *controlled-U^{2^j}* para cualquier entero j y necesitamos preparar de manera eficiente el vector propio $|u_s\rangle$ con su correspondiente valor propio no trivial, o en su defecto una superposición de los correspondientes vectores propios.

Para el primer requisito, necesitamos utilizar la exponenciación modular, con esto podemos computar la secuencia de operaciones *controlled-U^{2^j}* de esta manera:

$$|z\rangle\,|y\rangle \to |z\rangle\,U^{z_t 2^{t-1}}\cdots U^{z_1 2^0}\,|y\rangle = |z\rangle\,\left|x^{z_t 2^{t-1}}\cdots x^{z_1 2^0} y \pmod N\right\rangle = |z\rangle\,|x^z y \pmod N\rangle$$

Así la idea básica es aplicar la secuencia de operaciones *controlled-U^{2^j}* como la multiplicación del contenido de un segundo registro mediante exponenciación modular calculando de manera inversa la función $x^z(\text{mod } N)$, ya que

$$x^z(\text{mod } N) = \left(x^{z_t 2^{t-1}} \pmod N\right)\cdots\left(x^{z_1 2^0} \pmod N\right)$$

tomando $t = 2L + 1 + (\log(2 + 1/2\epsilon)) = O(L)$, para computar esta secuencia de operaciones necesitamos $O(L^3)$ operaciones.

Para el segundo requisito, si r es el orden de x en $\mathbb{Z}_N^*$, veamos que los siguientes vectores $|u_s\rangle$ son vectores propios de U. Para ello tomamos:

$$U\,|u_0\rangle = \frac{1}{\sqrt{r}}(|x \pmod N\rangle + \left|x^2 \pmod N\right\rangle + \cdots + |x^r \pmod N\rangle)$$

$$= \frac{1}{\sqrt{r}}(|x \pmod N\rangle + \left|x^2 \pmod N\right\rangle\cdots\left|x^{r-1} \pmod N\right\rangle + 1) = |u_0\rangle$$

donde el valor propio es 1. Ahora definimos el valor propio $\omega_r = e^{2\pi i/r}$ y tomamos lo siguiente:

$$U\,|u_1\rangle = \frac{1}{\sqrt{r}}(|x \pmod N\rangle + \omega_r^{-1}\left|x^2 \pmod N\right\rangle + \cdots + \omega_r^{-(r-1)}\,|x^r \pmod N\rangle)$$

$$= \frac{\omega_r}{\sqrt{r}}(\omega_r^{-1}\,|x \pmod N\rangle + \omega_r^{-2}\left|x^2 \pmod N\right\rangle + \cdots + \omega_r^{-r}) = \omega_r\,|u_1\rangle$$

En general, los valores propios vienen definidos por:

$$|u_s\rangle = \frac{1}{r}\sum_{k=0}^{r-1} e^{-2\pi isk/r}\,|x \pmod N\rangle$$

para $0 \le s \le r-1$, por tanto:

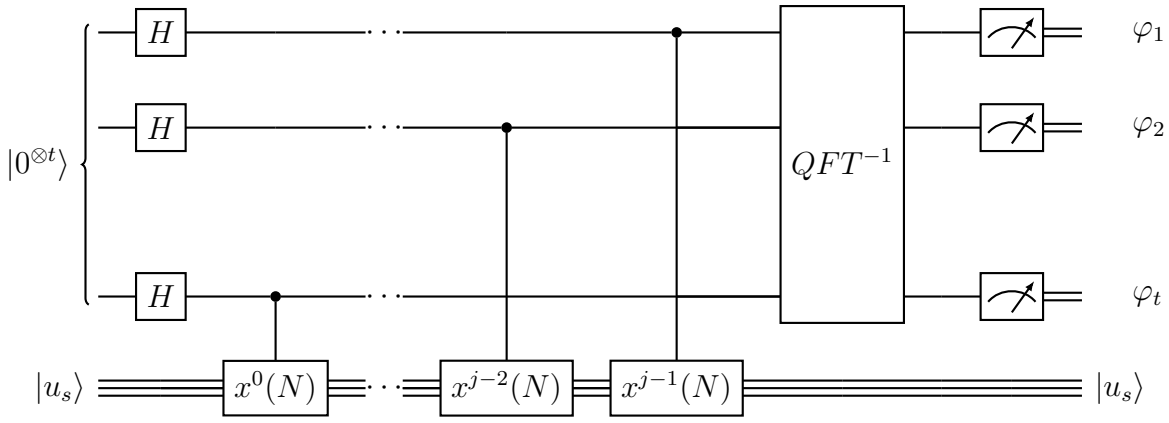
$$U\,|u_s\rangle = \frac{1}{r}\sum_{k=0}^{r-1} e^{-2\pi isk/r}\left|x^{k+1} \pmod N\right\rangle = e^{2\pi is/r}\,|u_s\rangle$$

Por último, para preparar de manera adecuada los vectores propios $|u_s\rangle$ es necesario utilizar un truco basado en la siguiente observación:

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}e^{-2\pi isk/r}\,|x \pmod N\rangle$$

$$\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}e^{-2\pi isk/r}e^{2\pi i\cdot 0\cdot k/r}\,|x \pmod N\rangle = \frac{1}{r}\sum_{k=0}^{r-1}r\delta_{k0}\,\left|x^k \pmod N\right\rangle = |1\rangle$$

Así, para utilizar el algoritmo de estimación de fase, tomamos $t = 2L + 1 + (\log(2 + 1/2\epsilon))$ qubits para el primer registro, tomando el segundo registro inicializado en $|1\rangle$, y para un s entre 0 y r-1 obtenemos la estimación de fase $\phi \approx s/r$ con una precisión de $2L + 1$ bits y con una probabilidad de éxito de al menos (1 - ϵ)/r. El algoritmo se representa con el siguiente circuito:



Donde $x^{j-1}(N)$ equivale a $x^{j-1} \pmod N$.

3.2.2 Las fracciones continuas

Para terminar el algoritmo del cálculo del orden transformado al algoritmo de estimación de fase, necesitamos obtener el valor r a partir del resultado obtenido anteriormente $\varphi \approx s/r$. Conocemos que φ nos es dado con t bits de precisión y que a priori es un número racional al cual le podemos aplicar el algoritmo de las fracciones continuas para obtener r. Hay que tener en cuenta que s no se conoce durante el cálculo, por tanto se tendrá que repetir el algoritmo para los distintos valores de s para encontrar r con una probabilidad alta de éxito.

Sea una fracción continua simple y finita definida por el conjunto de enteros positivos $a_0, \cdots, a_N$ tal que:

$$[a_0, \cdots, a_N] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\cdots + \cfrac{1}{a_N}}}}$$

podemos definir de manera inductiva:

$$\begin{array}{lll} s_0 = a_0 & s_1 = a_1 a_0 + 1 & s_n = a_n s_{n-1} + s_{n-2} \\ r_0 = 1 & r_1 = a_1 & r_n = a_n r_{n-1} + r_{n-2} \end{array}$$

donde

$$[a_0, \cdots, a_N] = \frac{s_n}{r_n}$$

Así, aplicamos el algoritmo de las fracciones continuas para hallar la expansión en fracciones continuas de φ tal que utilicemos el siguiente teorema:

Teorema 2 *Sea s/r un número racional tal que:*

$$\mid \frac{s}{r} - \varphi \mid \leq \frac{1}{2r^2}$$

entonces la fracción continua de φ converge a s/r.

Como φ es una aproximación de s/r con una precisión de $2L + 1$ bits tenemos que:

$$\mid \frac{s}{r} - \varphi \mid \leq 2^{-2L-1} \leq \frac{1}{2r^2}$$

para $r \leq N \leq 2^L$, por tanto podemos aplicar el teorema.
Resumiendo, podemos llegar mediante el algoritmo de fracciones continuas a s_n/r_n sin factores comunes tal que $s_n/r_n = s/r$, donde r es el orden de x en $\mathbb{Z}_N^*$

Algoritmo de cálculo de orden

Inputs: Una operación lineal $U_{x,N}$ tal que $|j\rangle |k\rangle \rightarrow |j\rangle |x^j k (mod N)\rangle$, para x co-primo con N, $t = 2L + 1 + (\log(2 + 1/2\epsilon)$ qubits que inicialmente están en el estado $|0\rangle$ y L qubits que inicialmente están en estado $|1\rangle$.

Outputs: El menor entero $r > 0$ tal que $x^r \equiv 1 \pmod{N}$.

Ejecuciones: $O(L^3)$ operaciones.

Procedimiento:

- Estado inicial $|0\rangle |1\rangle$

- Aplicamos superposición al estado inicial y aplicamos la operación lineal $U_{x,N}$:

$$\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle \left|x^k (mod N)\right\rangle \approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{k=0}^{2^t-1} e^{2\pi isk/r} |k\rangle |u_s\rangle$$

- Aplicamos la transformada inversa de fourier:

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \left|\hat{s/r}\right\rangle |u_s\rangle$$

- Medimos el primer registro y obtenemos $\hat{s/r}$
- Aplicamos el algoritmo de las fracciones continuas y obtenemos r.

3.3 Factorización

La crucial observación de Shor fue que hay un algoritmo cuántico eficiente para resolver el problema del cálculo de orden y que la factorización se puede transformar en este problema, por tanto tener un algoritmo eficiente para calcular el orden es equivalente a tener un algoritmo eficiente para la factorización.

Para transformar el problema debemos proceder en dos pasos. El primer paso es ver que podemos calcular un factor de N si podemos encontrar una solución no trivial $x \neq \pm 1 \pmod{N}$ a la ecuación $x^2 \equiv 1 \pmod{N}$. El segundo paso es ver que un y elegido al azar y co-primo con N, es muy probable que tenga orden r par tal que $y^{r/2} \neq \pm 1 \pmod{N}$ y tal que $x \equiv y^{r/2} \pmod{N}$ es la solución no trivial de $x^2 \equiv 1 \pmod{N}$. Estos dos pasos se

basan en los siguientes teoremas.

Teorema 3 *Sea N un número no primo con L bits de longitud y sea $x \epsilon \{1, \cdots, N-1\}$ la solución no trivial de la ecuación $x^2 \equiv 1$ (mod N) tal que $x \neq 1$ (mod N) y $x \neq N$ - 1 = -1 (mod N). Entonces al menos uno de mcd(x-1, N) y mcd(x+1, N) es un factor no trivial de N y se puede calcular usando $O(L^3)$ operaciones.*

Demostración
Como $x^2 \equiv 1$ (mod N), entonces N divide a x^2-1 = (x+1)(x-1), por tanto N tiene algún factor común con (x+1) o con (x-1). Asumiendo que $1 < x < N$-1, tenemos que x-1 < x+1 < N, por tanto el factor común no puede ser el mismo N. Usando el algoritmo de euclides, podemos calcular mcd(x-1, N) y mcd(x+1, N) y obtener un factor no trivial de N utilizando $O(L^3)$ operaciones.

□

Proposición 2 *Si el mcd(a, p) = 1 entonces $a^{\phi(p)} \equiv 1 (mod\ p)$*

Demostración(La demostración la encontramos en el anexo 4, proposición 3)

Consideremos ahora el grupo multiplicativo

$$\mathbb{Z}_N^* = \{a \epsilon \mathbb{Z}_N : mcd(a, N) = 1\}$$

de todos los elementos de Z_N que son co-primos con N y que contiene $\phi(N)$ elementos. $\mathbb{Z}_N$ es un grupo cíclico que contiene $\phi(N)$ generadores, es decir que todos sus elementos son de la forma $g \epsilon \mathbb{Z}_N$ que genera a $\mathbb{Z}_N$, esto es que $\mathbb{Z}_N = \{g^n \mid n \epsilon \mathbb{N}\}$.

Teorema 4 *Sea p un número primo impar y $\alpha > 1$, entonces $\mathbb{Z}_{p^\alpha}^*$ es un grupo cíclico.*

Demostración(La demostración la encontramos en el anexo 4, en el teorema 8)

Lema 1 *Sea p un número primo impar y sea 2^d la mayor potencia de 2 que divide a $\phi(p^\alpha)$. Entonces con una probabilidad de exactamente 1/2, 2^d divide al orden módulo p^α de cualquier elemento de $\mathbb{Z}_{p^\alpha}^*$*

Demostración.(La demostración la encontramos en el anexo 4, en el lema 2

Teorema 5 *Supongamos que* $N = p_1^{\alpha_1} \cdots p_m^{\alpha_m}$ *es la factorización en números primos de un número entero, positivo, impar y compuesto. Sea* x *elegido aleatoriamente de* $\mathbb{Z}_N^*$ *y sea* r *el orden de* x *módulo* N*. Entonces*

$$p(r \text{ es par y } x^{r/2} \not\equiv -1(mod\ N)) \geq 1 - \frac{1}{2^m}$$

Demostración.
Veamos que

$$p(r \text{ es impar y } x^{r/2} \equiv -1(\text{mod } N)) \leq \frac{1}{2^m}$$

Por el teorema chino de los restos, sabemos que elegir un elemento aleatorio x de $\mathbb{Z}_N^*$ es equivalente a tomar un elemnto x_j para un $\mathbb{Z}_{p_j^{\alpha_j}}^*$ aleatorio tal que $x = x_j(\text{mod } p_j^{\alpha_j})$ para cada j. Sea r_j el orden de x_j módulo $p_j^{\alpha_j}$. Sea 2^{d_j} la mayor potencia de dos que divide a r_j y 2^d la mayor potencia de dos que divide a r. Ahora tenemos que ver que para tener r impar o $x^{r/2} \equiv -1(\text{mod } N))$ es necesario que el valor de d_j sea el mismo para todos los valores de j. Este resultado se obtiene a partir del lema anterior, la probabilidad de que esto ocurra es a lo sumo $1/2^m$.
Consideremos dos casos, el primer caso es cuando r es impar. Es sencillo observar que $r_j \mid r$ para cada j, por tanto r_j es impar, y así $2_j^d = 1$ y $d_j = 0$ para cada valor de j.
El segundo caso es cuando r es par y $x^{r/2} \equiv -1(\text{mod } N))$. Entonces $x^{r/2} \equiv -1(\text{mod } p_j^{\alpha_j}))$, por tanto $r_j \nmid (r/2)$. Por tanto como $r_j \mid r$ tenemos que $d_j = d$ para todos los valores de j.

□

Combinando el teorema 3 y el 5 podemos dar con un algoritmo que con una alta probabilidad nos devuelva un factor no trivial de cualquier número compuesto N. Todos los pasos del algoritmo pueden ser computados de manera clásica y eficiente excepto el algoritmo de cálculo del orden, el cual debe ser computado de manera cuántica para ser eficiente. El algoritmo se resume de la siguiente forma.

3.3.1 Algoritmo de factorización

Inputs: Un número compuesto N.

Outputs: Un factor no trivial de N.
Ejecuciones: $O((\log N)^3)$ operaciones. Finaliza con éxito con una probabilidad $O(1)$.

Procedimiento:

- **Paso 1.** Si N es par, devolvemos el factor 2.
- **Paso 2.** Escogemos un entero aleatorio x tal que $1 < x < N$. Calculamos $d = \text{mcd}(x, N)$:
  - Si $d > 1$, d es un factor de N.
  - Si $d = 1$, procedemos al siguiente paso.
- **Paso 3.** Utilizamos el algoritmo de cálculo del orden r de x módulo N, es decir, calculamos el menor r tal que:

$$x^r \equiv 1 \pmod{N}$$

- **Paso 4.** Si r es par y $x^{r/2} \not\equiv -1((mod)\ N)$ entonces calculamos $d = \text{mcd}(x^{r/2} + 1, N)$ y $d' = \text{mcd}(x^{r/2} - 1, N)$. Y así obtenemos d y d' que son factores no triviales de N, en otro caso volvemos al paso 2.

El paso 1 devuelve el fator 2 si N es par en $O(1)$ operaciones. El paso 2 nos devuelve un elemento aleatorio x de $\{0, 1, 2, \cdots, N-1\}$. El paso 3 llama al algoritmo de cálculo del orden para encontrar el orden r de x módulo N. Por último, el paso 4 completa el algoritmo ya que el teorema 5 garantiza que con probabilidad de al menos $1/2$ r va a ser par y $x^{r/2} \not\equiv -1((mod)\ N)$, y con el teorema 3 se garantiza que $\text{mcd}(x^{r/2} - 1, N)$ o $\text{mcd}(x^{r/2} + 1, N)$ es un factor no trivial de N.

3.4 Implementación

Para dar un ejemplo de este algoritmo, se ha utilizado una cuenta en IBM Quantum Expirience ([16]). Aquí podemos encontrar gran cantidad de material para estudiar y desarrollar sobre el tema de la computación cuántica, ya que se pueden dibujar e implementar circuitos cuánticos. El lenguaje que se puede utilizar en el notebook de *jupyter* de la página es *python* y la librería

especializada en computación cuántica es *Qiskit*.

Se ha implementado un algoritmo para calcular los factores no triviales del número 15 con 3 qubits de precisión, 3 qubits para implimentar las operaciones *Controlled*-U, 1 qubit ancilla y 3 bits clásicos para almacenar la medición:

```
#número a factorizar
N = 15
#número de qubits de precisión
n = 3
#inicializamos el circuito para el algoritmo de Shor
qcShor = QuantumCircuit(4+n,n)
```

Figura 3: Iniciamos las variables

Definimos el método que computa el test de Miller Rabin para comprobar si es un número primo o no. Se elige $k = 40$ por ser el número óptimo de pasos para realizar el test, para más información (repositorio en GitHub y [18]):

```
def miller_rabin(n,k):
    if n == 2 or n == 3:
        return True
    if n % 2 == 0:
        return False
    r, s = 0, n - 1
    while s % 2 == 0:
        r += 1
        s //= 2
    for _ in range(k):
        a = random.randrange(2, n - 1)
        x = pow(a, s, n)
        if x == 1 or x == n - 1:
            continue
        for _ in range(r - 1):
            x = pow(x, 2, n)
            if x == n - 1:
                break
        else:
            return False
    return True
```

Figura 4: Método Miller-Rabin

Ahora procedemos al paso 1, comprobamos el número N es un número

primo o es un número par:

```
if miller_rabin(N,40):
    print("El número "+str(N)+" es primo")
if N % 2 == 0:
    print("Es un número par, el 2 es un factor no trivial")
else:
    print("procedemos al paso 2")
```

```
procedemos al paso 2
```

Figura 5: Método para comprobar el si el número es primo o par

En el paso 2, elegimos un número aleatorio entre $\{2, N-1\}$ y calculamos $d = \text{mcd}(x, N)$:

```
def gcd(a, b):
    if b > a:
        a, b = b, a
    while b > 0:
        a = a % b
        a, b = b, a
    return a
x = random.randrange(2, N - 1)
print("x = "+str(x))
d = gcd(x,N)
print("mcd("+str(x)+","+str(N)+") = "+str(d))
if d > 1:
    print(str(d) + " es un factor no trivial de "+str(N))
if d == 1:
    print("procedemos al paso 3")
```

```
x = 7
mcd(7,15) = 1
procedemos al paso 3
```

Figura 6: Elección del número aleatorio y primo con 15

En el paso 3 llegamos a la parte cuántica del algorítmo, el cálculo del orden r de x mod N, o lo que es lo mismo, encontrar el periodo de la función $f(r) = x^r (\text{mod } N)$. Veamos un ejemplo de cuál es el orden de 7 mod 15 observando la gráfica de la función $f(r) = 7^r (\text{mod } 15)$:

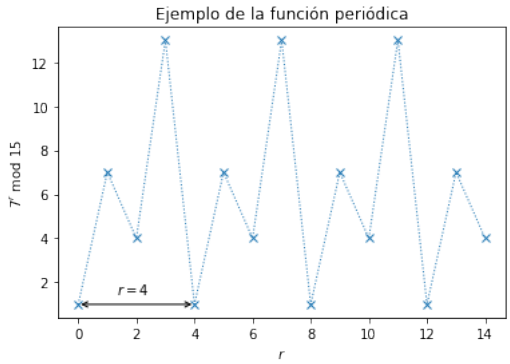


Figura 7: Grafica de la función $f(r) = 7^r (\text{mod } 15)$

Donde podemos observar que en este caso el período de la función o el orden de 7 mod 15 es $r = 4$. Ahora implementamos las operaciones C-U según el anexo 4:

```
def c_amod15(x, power):
    U = QuantumCircuit(4)
    for iteration in range(power):
        if x in [2,13]:
            U.swap(0,1)
            U.swap(1,2)
            U.swap(2,3)
        if x in [7,8]:
            U.swap(2,3)
            U.swap(1,2)
            U.swap(0,1)
        if x == 11:
            U.swap(1,3)
            U.swap(0,2)
        if x in [7,11,13]:
            for q in range(4):
                U.x(q)
    U = U.to_gate()
    U.name = "%i^%i mod 15" % (x, power)
    c_U = U.control()
    return c_U
```

Figura 8: Función de cálculo de las operaciones C-U

La siguiente función es la transformada cuántica de Fourier inversa:

```python
def qft_dagger(n):
    qc = QuantumCircuit(n)
    for qubit in range(n//2):
        qc.swap(qubit, n-qubit-1)
    for j in range(n):
        for m in range(j):
            qc.cu1(-np.pi/float(2**(j-m)), m, j)
        qc.h(j)
    qc.name = "QFT†"
    return qc
```

Figura 9: Función de la transformada cuántica de Fourier

Ahora, aplicamos la transformación Hadamard a los 3 primeros qubits e inicializamos el bit ancilla en el estado $|1\rangle$:

```python
for q in range(n):
    qcShor.h(q)
qcShor.x(3+n)
qcShor.draw("mpl")
```

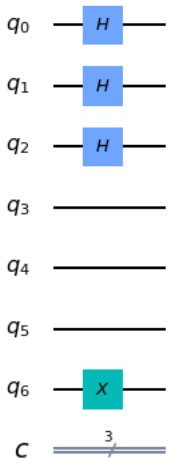


Figura 10: Función de la transformada cuántica de Fourier

Ahora aplicamos las operaciones C-U y la transformada inversa de Fourier, antes de ver el circuito, veamos un ejemplo de lo que se esconde en la operación C-U:

```
U = QuantumCircuit(4)
for iteration in range(2**1):
    U.swap(2,3)
    U.swap(1,2)
    U.swap(0,1)
    for q in range(4):
        U.x(q)
U.draw()
```

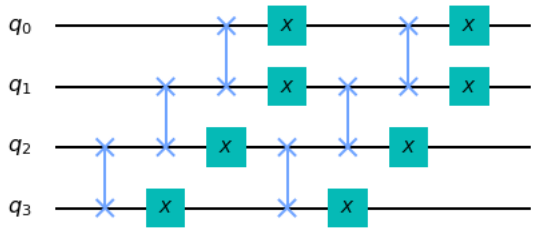


Figura 11: Operación C-U para $7^1((mod)\ 15)$

Después de aplicar la transformada inversa de Fourier se miden los tres primeros quibits sobre los tres bits clásicos:

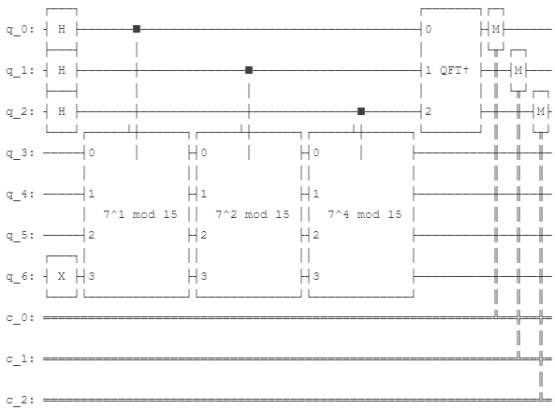


Figura 12: Operación C-U para $7^1((mod)\ 15)$

En el simulador de IBM, se realizan 2048 pruebas para observar los distintos resultados medidos:

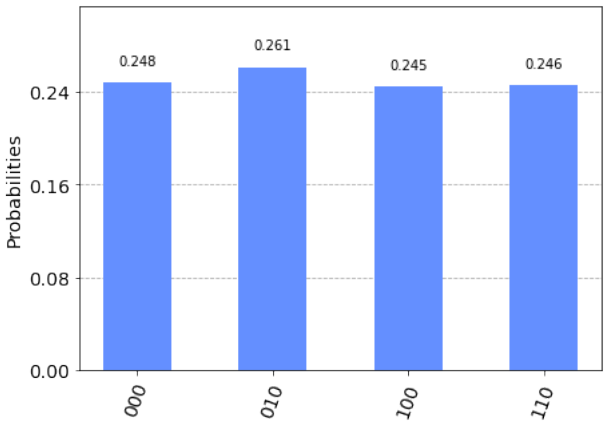


Figura 13: Histograma con el porcentaje de los resultados medidos

Utilizando la función .as_integer_ratio() de python podemos calcular los valores de las fracciones continuas para hallar los valores de las distintas fases:

```
rows, eigenvalues, r = [], [], []
for output in counts:
    decimal = int(output, 2)
    eigenvalue = decimal/(2**n)
    eigenvalues.append(eigenvalue)
    valueR = eigenvalue.as_integer_ratio()[1]
    r.append(valueR)
    rows.append(["%s(binario) = %i(decimal)" % (output, decimal),
                 "phase = %i/%i = %.2f" % (decimal, 2**n, eigenvalue), r])

for row in rows:
    print(row)
```

```
['100(binario) = 4(decimal)', 'phase = 4/8 = 0.50', [2, 1, 4, 4]]
['000(binario) = 0(decimal)', 'phase = 0/8 = 0.00', [2, 1, 4, 4]]
['110(binario) = 6(decimal)', 'phase = 6/8 = 0.75', [2, 1, 4, 4]]
['010(binario) = 2(decimal)', 'phase = 2/8 = 0.25', [2, 1, 4, 4]]
```

Figura 14: Muestra de las cuatro distintas fases obtenidas utilizando la función .as_integer_ratio()

Pasamos al paso cuatro y comprobamos que el valor de r sea distinto de 0 y par, si es así, calculamos $d = \text{mcd}(7^{r/2} + 1, 15)$ y $d^{'} = \text{mcd}(7^{r/2}$ - 1, 15) y comprobamos que sean valores no triviales de 15.

```
factor_found = False
pos = 0
while not factor_found:
        s, r = eigenvalues[pos].as_integer_ratio() # phase = s/r
        print("Orden: r = %i" % r)
        print(eigenvalues[pos].as_integer_ratio())
        if eigenvalues[pos] != 0 and r//2 % 2 == 0:
            guesses = [gcd(x**(r//2)-1, 15), gcd(x**(r//2)+1, 15)]
            print("Factores supuestos: %i and %i" % (guesses[0], guesses[1]))
            for guess in guesses:
                if guess != 1 and (N % guess) == 0: #Comprobamos los factores
                    print("*** factor no trivial: %i ***" % guess)
                    factor_found = True
        pos = pos + 1
```

```
Orden: r = 2
(1, 2)
Orden: r = 1
(0, 1)
Orden: r = 4
(3, 4)
Factores supuestos: 3 and 5
*** factor no trivial: 3 ***
*** factor no trivial: 5 ***
```

Figura 15: Resultados finales

Donde podemos observar que para el valor de fase 3/4 con $r = 4$, obtenemos los factores no triviales de 15, el 3 y el 5.

4 Conclusión

A partir de esta introducción hemos podido recorrer desde el principio el inicio de una rama muy interesante de esta ciencia. Desde una introducción de las bases que la sustentan, hasta uno de los algoritmos más importantes. He podido aprender desde el funcionamiento de los qubits y su entrelazamiento, pasando por las bases teóricas matemáticas, hasta poder programar en IBM Quantum experience circuitos cuánticos y participar en su challenge de su cuarto aniversario.

La experiencia de realizarlo, muy nutritiva, y las recientes noticias, y no tan recientes, sobre la computación cuántica, hacen que el interés por ampliar los conocimientos incremente. Noticias como la de 2001, en la que el grupo de IBM de computación cuántica liderado por Isaac L. Chuang logró demostrar experimentalmente con una molécula de fluor el algoritmo propuesto por shor, los ordenadores cuánticos fabricados por distintas empresas como Google o IMB, los esfuerzos de las primeras potencias mundiales por conseguir la supremacía de esta tecnología, Google consiguió resolver un problema en unos 200 segundos, cuando con supercomputadoras clásicas se estimaba que se habría conseguido resolver en 10000 años, o la noticia más reciente, en la que China a conseguido transmitir un mensaje cifrado, utilizando la computación cuántica, imposible de descifrar entre dos estaciones separadas por 1120 kilómetros.

Todo esto, sumado a la importancia científica y gubernamental otorgada a algoritmos como el propuesto por Shor, nos proporciona la curiosidad necesaria para seguir ampliando el conocimiento sobre esta materia.

Anexo 1: Nociones básicas

Anexo que referencia algunas de las nociones básicas utilizadas durante este trabajo.

Espacio de Hilbert

Esta sección tratara sin profundizar sobre la noción de espacio de Hilbert.

Definición 7 *(Semi-norma, espacio semi-normado). Sea* $\mathbb{K} = \mathbb{R}$ *o* $\mathbb{K} = \mathbb{C}$ *el cuerpo de los números reales o de los números complejos. Sea* ν *un espacio vectorial sobre* $\mathbb{K}$*. Decimos que la función* $\|.\| : \nu \to [0, \infty)$ *es una semi-norma en* ν *si:*

- $\|x + y| \leq \|x\| + \|y\|, \forall x, y \epsilon \nu$ *(desigualdad triangular);*
- $\|\lambda x\| =\mid \lambda \mid \|x\|, \forall \lambda \epsilon \mathbb{K}, x \epsilon \nu$ *(Homogeneidad).*

En tal caso, decimos que $(\nu, \|.\|)$ *es un espacio vectorial semi-normado.*

Definición 8 *(Norma, espacio normado). Si la semi-norma cumple además*

$$\|x\| = 0 \Leftrightarrow x = 0,$$

entonces se dice que es una norma y que ν *es un espacio normado.*

Definición 9 *(Sucesión de Cauchy. Espacio de Banach). Una sucesión* $\{x_n\}_{n \epsilon \mathbb{N}}$ *es una espacio normado* $(\nu, \|.\|)$*se dice que es una Sucesión de Cauchy si para todo* $\epsilon > 0$ *existe* $N \epsilon \mathbb{N}$ *tal que* $\forall n, m \epsilon \mathbb{N}$*,* $n, m \neq N$ *se cumple*

$$d(x_n, x_m) = \|x_n - x_m\| < \epsilon$$

El espacio $(\nu, \|.\|)$ *se dice que es un espacio de Banach si toda sucesión de Cauchy en dicho espacio es convergente; es decir, existe* $x \epsilon \nu$ *tal que*

$$\lim_n x_n = x \Leftrightarrow \lim_n \|x_n - x\| = 0.$$

También se dice que $(\nu, \|.\|)$ *es un espacio de Banach si su espacio métrico asociado es completo.*

Definición 10 *(Producto escalar). Sea* h *un espacio vectorial. Un producto escalar en* H *es una función de* $H \times H$ *en* $\mathbb{K}$ *que a cada* (x, y) $\epsilon H \times H$ *le asigna el valor* $\langle x, y \rangle$ *que tiene las siguientes propiedades:*

1. $\langle x, y\rangle = \overline{\langle x, y\rangle}, \forall x, y \epsilon H.$
2. $\langle x + y, z\rangle = \langle x, z\rangle + \langle y, z\rangle, \forall x, y, z \epsilon H.$
3. $\langle \alpha x, y\rangle = \alpha\langle x, y\rangle \forall x, y, z \epsilon H, \forall \alpha \epsilon \mathbb{K}.$
4. $\langle x, x\rangle \neq 0, \forall x \epsilon H.$
5. $\langle x, x\rangle = 0 \rightarrow x = 0.$

En tal caso decimos que H es un espacio vectorial con producto escalar.

Definición 11 *(Espacio de Hilbert). Si H es un espacio con producto escalar tal que la norma asociada es un espacio de Banach, entonces decimos que es un espacio de Hilbert.*

Notación

Definición 12 *(Notación de Dirac). Dado un espacio de Hilbert $\mathbb{H} = \mathbb{C}^n$, con una cantidad $\psi \epsilon \mathbb{H}$, denotada por $|\psi\rangle$ (ket), es un vector y lo podemos considerar como un vector columna. Dada una cantidad $\phi \epsilon \mathbb{H}^*$, denotado por $\langle\phi|$ (bra), es un vector en el espacio dual y se puede considerar como un vector fila que es la conjugada traspuesta de $\phi \epsilon \mathbb{H}$.*

Nota: Así, una expresión como $\langle\phi||\psi\rangle$ es un producto interno en el espacio de Hilbert. En este trabajo los espacios de Hilbert que usaremos serán de la forma $(\mathbb{C}^2)^{\otimes q}$, donde q es un entero dado.

Definición 13 *La base estandar de $\mathbb{C}^2$ es denotada por $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ y $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$. La base estandar en $((\mathbb{C}^2)^{\otimes q}$ contiene 2^q elementos y se denota por $|0\rangle_q$, $|1\rangle_q$, ..., $|2^q - 1\rangle_q$*

Producto Tensor

Definición 14 *(Producto tensorial). Sean V, W espacios vectoriales sobre un cuerpo $\mathbb{K}$ con bases $e_1, \ldots, e_m$ y $f_1, \ldots, f_n$ respectivamente. El producto tensorial es otro espacio vectorial sobre $\mathbb{K}$ con dimensión $m \times n$. El espacio producto tensorial esta formado por una operación bilineal $\otimes$: $V \times W \rightarrow V \otimes W$. EL espacio vectorial $V \otimes W$ tiene como base $e_i \otimes f_j$ $\forall i = 1, \cdots, m$, $\forall j = 1, \cdots, n$.*

Nota: Si los espacios vectoriales son espacios de Hilbert del tipo $\mathbb{H} = \mathbb{C}^n$ y elegimos la base estándar en los espacios vectoriales de origen, entonces el producto vectorial no es otro que el producto de Kronecker, que es la generalización del producto exterior.

Definición 15 *(Producto de Kronecker): Sean* $A \epsilon \mathbb{C}^{m\times n}$ *y* $B \epsilon \mathbb{C}^{p\times q}$*, el priducto de Kronecker* $A \otimes B$ *es la matriz* $D \epsilon \mathbb{C}^{mp\times nq}$ *definida como:*

$$D := A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ a_{21}B & \cdots & a_{2n}B \\ & \cdots & \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}$$

Si elegimos las bases estándar de $\mathbb{C}^{m\times n}$ *y* $\mathbb{C}^{p\times q}$*, entonces la operación bilineal* $\otimes$ *del producto tensorial* $\mathbb{C}^{m\times n} \otimes \mathbb{C}^{p\times q}$

Nota: En este trabajo se trabaja con espacios de Hilbert complejos de la forma $\mathbb{C}^n$. Abusando de la notación del producto tensorial, se usará para referirse al producto de Kronecker.

Anexo 2: Convolución Discreta

Este anexo busca completar la sección 2.1.2 de la multiplicación de dos polinomios.

Definición 16 *Dados dos vectores $a, b \in \mathbb{C}^N$, la convolución de estos dos vectores es un vector $a * b \in \mathbb{C}^N$ cuya l-entrada viene definada por*

$$(a * b)_l = \frac{1}{\sqrt{N}} \left(\sum_{j=0}^{N-1} a_j b_{l-j} + \sum_{j=0}^{N-1} a_j b_{N+l-j} \right)$$

Dado $m \in \mathbb{Z}$, denotemos por $m \bmod n$ el resto al dividir el número m entre n. Con esta notación vamos a escribir de manera más breve la convolución:

$$(a * b)_l = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j b_{(l-j) mod N}$$

Si de la definición tomamos $N = 2d + 1$ (el número de coeficientes no nulos de $p \cdot q$), hacemos el producto $(d + 1)$-dimensional anterior de los vectores a y b, y tomamos del producto polinomial

$$(p \cdot q)(x) = \left(\sum_{j=0}^{d} a_j x^j \right) \left(\sum_{k=0}^{d} a_k x^k \right) = \sum_{l=0}^{2d} \left(\sum_{j=0}^{2d} a_j b_{l-j} \right) x^l$$

el coeficiente

$$c_l = \sum_{j=0}^{2d} a_j b_{l-j}$$

veremos que el producto $p \cdot q$ es proporcional a las entradas de la convolución de $c_l = \sqrt{N}(a * b)_l$. Por tanto, a partir de aquí, es sencillo observar que los coeficientes de Fourier de la convolución coincide con el producto de los coeficientes de Fourier de a y de b. Esto nos lo da el teorema de convolución.

Teorema 6 *(de convolución). Sean a, $b \in \mathbb{C}^n$. Entonces*

$$F_n(a * b) = (F_n(a)) \cdot (F_n(b)),$$

o lo que es lo mismo

$$\widehat{(a * b)} = \hat{a} \cdot \hat{b}$$

Demostración. Dado un $j \in \{0, ..., n-1\}$, vamos a observar que las j-ésimas componentes de estos vectores son iguales entre sí. Vamos a calcular la j-ésima componente del lado izquierdo:

$$(F_N(a * b))_j = \sum_{k=0}^{n-1} (F_N)_{j,k}(a \cdot b)_j = \sum_{k=0}^{n-1} \omega_n^{jk} \left(\sum_{q=0}^{k} a_{k-q} b_q + \sum_{q=k+1}^{n-1} a_{n+k-q} b_q \right)$$

aplicando las propiedades de las operaciones en $\mathbb{C}$ obtenemos:

$$(F_N(a * b))_j = \sum_{k=0}^{n-1} \sum_{q=0}^{k} \omega_n^{jk} a_{k-q} b_q + \sum_{k=0}^{n-1} \sum_{q=k+1}^{n-1} \omega_n^{jk} a_{n+k-q} b_q$$

$$= \sum_{q=0}^{n-1} \sum_{k=q}^{n-1} \omega_n^{jk} a_{k-q} b_q + \sum_{q=0}^{n-1} \sum_{k=0}^{q-1} \omega_n^{jk} a_{n+k-q} b_q$$

Ahora, hacemos un cambio de variable en el primer sumatorio de la siguiente forma:

$$s = k - q$$
$$k = s + q$$

donde k recorre de q a $n-1$ y s recorre de 0 a $n-q-1$. En el segundo sumatorio hacemos el cambio de la siguiente forma:

$$s = n + k - q$$
$$k = s + q - n$$

donde k recorre de 0 a $q-1$ y s recorre de $n-q$ a $n-1$. Así, obtenemos:

$$(F_N(a * b))_j = \sum_{q=0}^{n-1} \sum_{s=0}^{n-q-1} \omega_n^{js+jq} a_s b_q + \sum_{q=0}^{n-1} \sum_{s=n-q}^{n-1} \omega_n^{js+jq-jn} a_s b_q$$

Como $\omega_n^{-jn} = 1$, juntamos las dos sumas en una, separamos despues la s y la q y obtenemos:

$$(F_N(a * b))_j = \sum_{q=0}^{n-1} \sum_{s=0}^{n-q-1} \omega_n^{js+jq} a_s b_q = \sum_{s=0}^{n-1} \omega_n^{js} a_s \sum_{q=0}^{n-1} \omega_n^{jq} b_q$$

$$= \sum_{s=0}^{n-1} (F_n)_{j,s} a_s \sum_{q=0}^{n-1} (F_n)_{j,q} b_q = (F_n(a))_j (F_n(b))_j = ((F_n(a)) \cdot (F_n(b)))_j$$

Que es la igualdad a la que queriamos llegar. $\square$

Este resultado, inmediatamente nos sugiere el siguiente corolario, con el cual podremos hallar el algoritmo con el que computaremos el vector de coeficiente c_l:

corolario 2 *Sean* $a,\ b\ \epsilon\ \mathbb{C}^n$. *Entonces*

$$a * b = F_n^{-1}((F_n(a)) \cdot (F_n(b)))$$

Anexo 3: Reducción a la factorización

En este anexo completaremos la sección 3.3 de la factorización.

Para transformar el problema debemos proceder en dos pasos. El primer paso es ver que podemos calcular un factor de N si podemos encontrar una solución no trivial $x \neq \pm 1 \pmod{N}$ a la ecuación $x^2 \equiv 1 \pmod{N}$. El segundo paso es ver que un y elegido al azar y co-primo con N, es muy probable que tenga orden r par tal que $y^{r/2} \neq \pm 1 \pmod{N}$ y tal que $x \equiv y^{r/2} \pmod{N}$ es la solución no trivial de $x^2 \equiv 1 \pmod{N}$. Estos dos pasos se basan en los siguientes teoremas.

Teorema 7 *Sea N un número no primo con L bits de longitud y sea $x \epsilon \{1, cdots, N-1\}$ la solución no trivial de la ecuación $x^2 \equiv 1$ (mod N) tal que $x \neq 1$ (mod N) y $x \neq N$ - 1 = -1 (mod N). Entonces al menos uno de mcd(x-1, N) y mcd(x+1, N) es un factor no trivial de N y se puede calcular usando $O(L^3)$ operaciones.*

Demostración
Como $x^2 \equiv 1 \pmod{N}$, entonces N divide a x^2-1 = (x+1)(x-1), por tanto N tiene algún factor común con (x+1) o con (x-1). Asumiendo que $1 < x < N$-1, tenemos que x-1 < x+1 < N, por tanto el factor común no puede ser el mismo N. Usando el algoritmo de Euclides, podemos calcular mcd(x-1, N) y mcd(x+1, N) y obtener un factor no trivial de N utilizando $O(L^3)$ operaciones.

□

Consideremos ahora el grupo multiplicativo

$$\mathbb{Z}_N^* = \{a \epsilon \mathbb{Z}_N : mcd(a, N) = 1\}$$

de todos los elementos de Z_N que son co-primos con N y que contiene $\phi(N)$ elementos. $\mathbb{Z}_N$ es un grupo cíclico que contiene $\phi(N)$ generadores, es decir que todos sus elementos son de la forma $g \epsilon \mathbb{Z}_N$ que genera a $\mathbb{Z}_N$, esto es que $\mathbb{Z}_N = \{g^n \mid n \epsilon \mathbb{N}\}$.

Proposición 3 *Si el mcd(a, m) = 1 entonces $a^{\phi(m)} \equiv 1 (mod\ m)$*

Demostración.

- Si $m = p$ utilizamos que como p es primo, entonces para cada número natural $a > 0$ co-primo con p, $a^{p-1} \equiv 1 \pmod{p}$(enunciado del pequeño teorema de Fermat).

- Si $m = p^\alpha$ con p primo y $\alpha > 1$, supongamos que el enunciado es cierto para $p^{\alpha-1}$, entonces

$$a^{\phi(p^{\alpha-1})} = a^{p^{\alpha-1}-p^{\alpha-2}} \equiv 1(\text{mod } p^{\alpha-1})$$

entonces existe $b \epsilon \mathbb{N}: \ a^{\phi(p^{\alpha-1})} = 1 + p^{\alpha-1}b$; por tanto

$$a^{\phi(p^\alpha)} = (a^{\phi(p^{\alpha-1})})^p = (1+p^{\alpha-1}b)^p = 1 + pp^{\alpha-1}b + \cdots \equiv 1(\text{mod } p^\alpha).$$

Finalmente en el caso general $m = \prod_{i=1}^{r} p_i^{\alpha_i}$, ya que para cada i, $x^{\phi(p_i^{\alpha_i})} \equiv (\text{mod } p_i^{\alpha_i})$ y $\phi(p_i^{\alpha_i}) \mid \phi(m)$ tenemos que $x^{\phi(m)} \equiv 1(textrm(mod)\, p_i^{\alpha_i}$ para cada i, así el resultado sigue del Teorema del resto chino.

□

Teorema 8 *Sea p un número primo impar y $\alpha > 1$, entonces $\mathbb{Z}_{p^\alpha}^*$ es un grupo cíclico.*

Demostración
Sea $\mathbb{Z}_p^*$ un grupo cíclico y $g \epsilon \mathbb{Z}_N^*$ su generador, tenemos que g o $g + p$, con p un número primo impar, es el generador de $\mathbb{Z}_{p^\alpha}^*$. Para probar esto, tomamos:

- Si $g^{p-1} \not\equiv 1 \ (\text{mod } p^2)$, tomamos $g_0 := g$, o
- Si $g^{p-1} \equiv 1 \text{ mod } p^2$, tomamos $g_0 := g + p$ donde tenemos:

$$g_0^{p-1} = (g+p)^{p-1} = \sum_{k=0}^{p-1} \binom{p-1}{k} g^{n-k}p^k \equiv g^{p-1} + (p-1)pg^{p-2} \equiv 1 - pg^{p-2} \not\equiv 1(\text{mod } p^2)$$

Para probar que g_0 es un generador de $\mathbb{Z}_{p^2}^*$ vamos a utilizar la siguiente proposición:

Proposición 4 *Si el mcd(a, p) = 1 entonces $a^{\phi(p)} \equiv 1(mod\ p)$*

En los dos casos $g_0^{p-1} \not\equiv 1 \ (\text{mod } p^2)$, por tanto g_0 es un generador de $\mathbb{Z}_{p^2}^*$ ya que

$$g_0^{\phi(p^2)} = g_0^{p^2-p} \equiv 1(\text{mod } p^2)$$

También sabemos que $g_0^{p-1} \not\equiv 1(\text{mod } p^2)$ implica que $g_0^{p-1} = 1 + g_1 p$ con $\text{mcd}(g_1, p) = 1$. Por tanto para $\alpha \geq 3$ y para $p > 2^2$ tenemos:

$$g_0^{(p-1)p^{\alpha-2}} = (1+g_1p)^{p^{\alpha-2}} \equiv 1 + p^{\alpha-2}g_1p + p^{\alpha-2}\frac{p^{\alpha-2}-1}{2}g_1^2p^2 \equiv 1 + g_1p^{\alpha-1} \not\equiv 1(\text{mod} p^\alpha$$

Por tanto, $g_0^{(p-1)p^{\alpha-2}} \not\equiv 1(\text{mod } p^\alpha)$ y asi g_0 es un generador de $\mathbb{Z}_{p^\alpha}^*$ ya que

$$g_0^{\phi(p^\alpha)} = g_0^{p^\alpha - p^{\alpha-1}} \equiv 1(\text{mod } p^2)$$

□

Lema 2 *Sea p un número primo impar y sea 2^d la mayor potencia de 2 que divide a $\phi(p^\alpha)$. Entonces con una probabilidad de exactamente 1/2, 2^d divide al orden módulo p^α de cualquier elemento de $\mathbb{Z}^*_{p^\alpha}$*

Demostración.
Observemos que $\phi(p^\alpha) = p^\alpha - p^{\alpha-1}$ es un número par ya que p es impar y así $d \geq 1$. Por el teorema anterior, exite un generador g de $\mathbb{Z}^*_{p^\alpha}$, así cualquier elemento aleatorio puede ser escrito de la forma $g^k (\text{mod } p^\alpha)$ para cualquier $K \epsilon \{1, \cdots, \phi(p^\alpha)\}$. Sea r el orden de este elemento, podemos considerar dos casos.

- El primer caso es cuando k es impar. Para $g^{kr} \equiv 1 (\text{mod } p^\alpha)$ podemos deducir que $\phi(p^\alpha) \mid kr$ y así, $2^d \mid r$ ya que k es impar.
- El segundo caso es cuando k es par. Entonces

$$g^{k\phi(p^\alpha)/2} = \left(g^{\phi(p^\alpha)}\right)^{k/2} \equiv 1^{k/2} \equiv 1 (\text{mod } p^\alpha)$$

  Por tanto $r \mid \phi(p^\alpha)/2$ de donde deducimos que 2^d no divide a r.

Resumiendo, podemos dividir $\mathbb{Z}^*_{p^\alpha}$ en dos subconjuntos de igual tamaño, los que se pueden escribir como $g^k (\text{mod } p^\alpha)$ con k impar y $2^d \mid r$, donde r es el orden de g^k, y los que se pueden escribir como $g^k (\text{mod } p^\alpha)$ con k par y $2^d \nmid r$. Por tanto, con una probabilidad de 1/2 el entero 2^d divide el orden r de un elemento aleatorio de $\mathbb{Z}^*_{p^\alpha}$, y con probabilidad de 1/2 no

□

Teorema 9 *Supongamos que $N = p_1^{\alpha_1} \cdots p_m^{\alpha_m}$ es la factorización en números primos de un número entero, positivo, impar y compuesto. Sea x elegido aleatoriamente de $\mathbb{Z}^*_N$ y sea r el orden de x módulo N. Entonces*

$$p(r \text{ es par y } x^{r/2} \not\equiv -1 (mod\ N)) \geq 1 - \frac{1}{2^m}$$

Demostración.
Veamos que

$$p(r \text{ es impar y } x^{r/2} \equiv -1 (\text{mod } N)) \leq \frac{1}{2^m}$$

Por el teorema chino de los restos, sabemos que elegir un elemento aleatorio x de $\mathbb{Z}_N^*$ es equivalente a tomar un elemnto x_j para un $\mathbb{Z}^*_{p_j^{\alpha_j}}$ aleatorio tal que $x = x_j (\text{mod } p_j^{\alpha_j})$ para cada j. Sea r_j el orden de x_j módulo $p_j^{\alpha_j}$. Sea 2^{d_j} la mayor potencia de dos que divide a r_j y 2^d la mayor potencia de dos que divide a r. Ahora tenemos que ver que para tener r impar o $x^{r/2} \equiv -1(\text{mod } N))$ es necesario que el valor de d_j sea el mismo para todos los valores de j. Este resultado se obtiene a partir del lema anterior, la probabilidad de que esto ocurra es a lo sumo $1/2^m$.
Consideremos dos casos, el primer caso es cuando r es impar. Es sencillo observar que $r_j \mid r$ para cada j, por tanto r_j es impar, y así $2_j^d = 1$ y $d_j = 0$ para cada valor de j.
El segundo caso es cuando r es par y $x^{r/2} \equiv -1(\text{mod } N))$. Entonces $x^{r/2} \equiv -1(\text{mod } p_j^{\alpha_j}))$, por tanto $r_j \nmid (r/2)$. Por tanto como $r_j \mid r$ tenemos que $d_j = d$ para todos los valores de j.

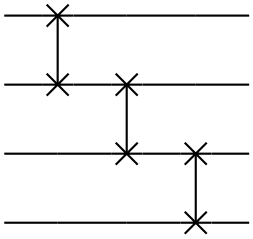
□

Anexo 4: Implementación

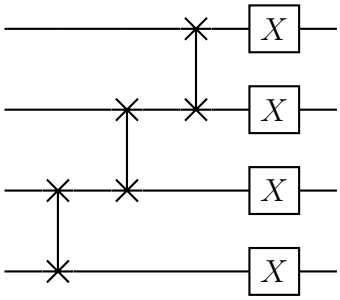
Este anexo busca completar la sección de implementación [3.4], como no es un objetivo de este trabajo se puede ampliar la información sobre la creación las operaciones *controlled* de sumas modulares en [15], [14] y [13].

Para implementar el algoritmo, comenzamos tomando $N = 15$. En la figura 3, podemos observar como se inicia el circuito cuántico con 3 qubits de precisión, 3 qubits para implimentar las operaciones *Controlled*-U, 1 qubit ancilla(bit extra que logra la computación reversible) y 3 bits clásicos para almacenar la medición. Después de comprobar que ni es primo ni es par, procedemos a elegir un número aleatorio coprimo con N tal que $a \, \epsilon \, [2, \, N-1]$. En este caso solo valen $\{2, 4, 7, 8, 11, 13\}$. Después se calcula el orden de $7^r (\text{mod } 15)$ para hallar que $r = 4$. Calculamos $7^{4/2} \pm 1 = \{48, 50\}$ donde obtenemos $d = mcd(48, 15) = 3$ y que $d^{'} = mcd(50, 15) = 5$.

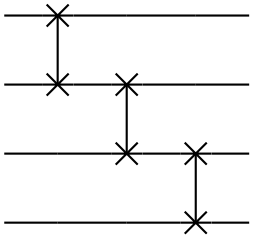
Para realizar las operaciones *Controlled*-U de la figura 8, se ha utilizado la aproximación de Kitaev [15]. Donde las operaciones $a^j \text{mod} N$ son las siguientes:



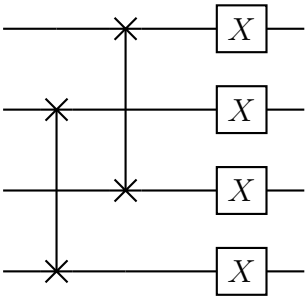
$2^1 \text{mod} 15$ con $r = 4$



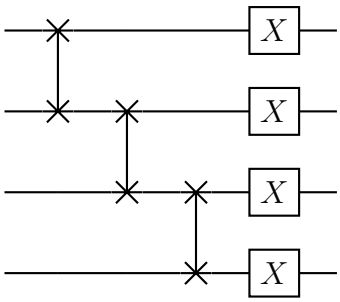
$7^1 \text{mod} 15$ con $r = 4$



$8^1 \text{mod} 15$ con $r = 4$



$11^1 \text{mod} 15$ con $r = 2$



$13^1 \text{mod} 15$ con $r = 4$

5 bibliografía

Referencias

[1] Michael A. Nielsen & Isaac L. Chuang: *Quantum Computation and Quantum Information*. Cambridge University Press (2010).

[2] Giacomo Nannicini: *An Introduction to Quantum Computing, Without the Physics*. IBM T.J. Watson, Yorktown Heights, NY, (2018); arXiv:1708.03684

[3] Ronald de Wolf: *Quantum Computing: Lecture Notes*. QuSoft, CWI and University of Amsterdam, (2018); arXiv:1907.09415

[4] Alastair Kay: *Tutorial on the Quantikz Package*. Royal Holloway University of London, Egham, Surrey, UK, (2019);arXiv:1809.03842

[5] Patrick J. Coles, Stephan Eidenbenz, Scott Pakin and others: *Quantum Algorithm Implementations for Beginners*. Los Alamos National Laboratory, Los Alamos, New Mexico, USA, (2018); arXiv:1804.03719

[6] Peter W. Shor: *Polynomial-Time Algorithms For Prime Factorization And Discrete Logarithms On A Quantum Computer*. Society for Industrial and Applied Mathematics, Journal on Computing, (1997); también en arXiv:quant-ph/9508027

[7] John Watrous: *John Watrous's Lecture Notes* `https://cs.uwaterloo.ca/~watrous/LectureNotes.html`; Sección *Introduction to Quantum Computing (notes from Winter 2006)*, lecturas de la 1 a la 11.(2006) [Consulta 17 de junio]

[8] Darío Coutiño Aquino y Egor Maximenko: *Convolución discreta cíclica*, http://esfm.egormaximenko.com/dft/discrete_cyclic_convolution_es.pdf

[9] José Juan Carreño Carreño: *Algoritmo de factorización de Shor. IV Seminario de Matemática Discreta*. http://www.dma.eui.upm.es/MatDis/Seminario4/AlgoritmoShor.pdf, Dpto. de Matemática Aplicada E. U. de Informática (U.P.M.).

[10] Ruben Dezeure & Manuel Schneider: *Shor's algorithm: Order finding and factorization*. https://qudev.phys.ethz.ch/static/content/courses/QSIT11/presentations/QSIT-ShorTheory.pdf, Department of Muster Second row.

[11] Teo Mora: *Carmichael*. http://www.dima.unige.it/~morafe/MaterialeCTC/Carmichael.pdf, Dipartimento di Matematica, Universita' di Genova.

[12] Donald E. Knuth: *The art of computer programming. Volume 2, Seminumerical Algorithms, Third edition*. Adison Wesley Longman, (1998).

[13] Stéphane Beauregard: *Circuit for Shor's algorithm using 2n+3 qubits*, (2003); arXiv:quant-ph/0205095.

[14] Christof Zalka: *Fast versions of Shor's quantum factoring algorithm*, (2008); arXiv:quant-ph/9806084.

[15] T. Monz, D. Nigg and others: *Realization of a scalable Shor algorithm* , (2015); arXiv:1507.08852.

[16] Héctor Abraham and AduOffei and Ismail Yunus Akhalwaya and others (ver todos en BibTeX file): *Qiskit: An Open-source Framework for Quantum Computing*. (2019); qiskit.org/

[17] Manuel Bello Hernández: *Apuntes de Análisis Real y Funcional*, Universidad de La Rioja, (2018).

[18] Juan Luis Varona: *Recorridos por la teoría de números*, Electolibris y la Real Sociedad Matemática Española; 2014.